

libDSP Reference Manual

4.9.2

Generated by Doxygen 1.3.6

Tue Mar 2 19:56:33 2004

Contents

1	libDSP Main Page	1
2	libDSP Hierarchical Index	3
2.1	libDSP Class Hierarchy	3
3	libDSP Class Index	5
3.1	libDSP Class List	5
4	libDSP File Index	7
4.1	libDSP File List	7
5	libDSP Class Documentation	9
5.1	_sDCplx Struct Reference	9
5.2	_sDPolar Struct Reference	10
5.3	_sSCplx Struct Reference	11
5.4	_sSPolar Struct Reference	12
5.5	_uDCoord Union Reference	13
5.6	_uSCoord Union Reference	14
5.7	clAlloc Class Reference	15
5.8	clCondition Class Reference	22
5.9	clDSPAlloc Class Reference	24
5.10	clDSPOp Class Reference	25
5.11	clDSPVector< TDSPVector_t > Class Template Reference	111
5.12	clDynThreads< TThreads > Class Template Reference	118
5.13	clDynThreads< TThreads >::_stParams2 Struct Reference	120
5.14	clDynThreadsBase Class Reference	121
5.15	clDynThreadsBase::_stParams Struct Reference	123

5.16	clException Class Reference	124
5.17	clFFFTDecimator Class Reference	126
5.18	clFFFTInterpolator Class Reference	128
5.19	clFFFTMultiRate Class Reference	130
5.20	clFilter Class Reference	133
5.21	clFIRDecimator Class Reference	144
5.22	clFIRInterpolator Class Reference	146
5.23	clFIRMultiRate Class Reference	148
5.24	clFlipBand Class Reference	151
5.25	clHankel Class Reference	154
5.26	clIIRCascade Class Reference	159
5.27	clIIRDecimator Class Reference	162
5.28	clIIRInterpolator Class Reference	164
5.29	clIIRMultiRate Class Reference	166
5.30	clMutex Class Reference	168
5.31	clPthCond Class Reference	170
5.32	clPthMutex Class Reference	171
5.33	clReBuffer Class Reference	172
5.34	clReBufferT< TReBuffer_t > Class Template Reference	175
5.35	clRecDecimator Class Reference	180
5.36	clRecInterpolator Class Reference	185
5.37	clRWLock Class Reference	190
5.38	clSemaphore Class Reference	193
5.39	clTransform4 Class Reference	196
5.40	clTransform8 Class Reference	205
5.41	clTransformS Class Reference	209
5.42	clUserThreads Class Reference	213
6	libDSP File Documentation	215
6.1	Alloc.hh File Reference	215
6.2	cdspop.cc File Reference	216
6.3	Compilers.hh File Reference	274
6.4	Condition.hh File Reference	275
6.5	DSPConfig.hh File Reference	276

6.6	DSPOp.cc File Reference	277
6.7	dspop.h File Reference	278
6.8	DSPOp.hh File Reference	327
6.9	dsptypes.h File Reference	328
6.10	DSPVector.hh File Reference	331
6.11	DynThreads.cc File Reference	332
6.12	DynThreads.hh File Reference	333
6.13	Exception.hh File Reference	334
6.14	FFTDecimator.cc File Reference	335
6.15	FFTDecimator.hh File Reference	336
6.16	FFTInterpolator.cc File Reference	337
6.17	FFTInterpolator.hh File Reference	338
6.18	FFTMultiRate.cc File Reference	339
6.19	FFTMultiRate.hh File Reference	340
6.20	Filter.cc File Reference	341
6.21	Filter.hh File Reference	342
6.22	FIRDecimator.cc File Reference	343
6.23	FIRDecimator.hh File Reference	344
6.24	FIRInterpolator.cc File Reference	345
6.25	FIRInterpolator.hh File Reference	346
6.26	FIRMultiRate.cc File Reference	347
6.27	FIRMultiRate.hh File Reference	348
6.28	FlipBand.cc File Reference	349
6.29	FlipBand.hh File Reference	350
6.30	Hankel.cc File Reference	351
6.31	Hankel.hh File Reference	353
6.32	IIRCascade.cc File Reference	354
6.33	IIRCascade.hh File Reference	355
6.34	IIRDecimator.cc File Reference	356
6.35	IIRDecimator.hh File Reference	357
6.36	IIRInterpolator.cc File Reference	358
6.37	IIRInterpolator.hh File Reference	359
6.38	IIRMultiRate.cc File Reference	360
6.39	IIRMultiRate.hh File Reference	361

6.40	Mutex.hh File Reference	362
6.41	PthCond.hh File Reference	363
6.42	PthMutex.hh File Reference	364
6.43	ReBuffer.cc File Reference	365
6.44	ReBuffer.hh File Reference	366
6.45	ReBufferT.hh File Reference	367
6.46	RecDecimator.cc File Reference	368
6.47	RecDecimator.hh File Reference	369
6.48	RecInterpolator.cc File Reference	370
6.49	RecInterpolator.hh File Reference	371
6.50	RWLock.hh File Reference	372
6.51	Semaphore.hh File Reference	373
6.52	Test.cc File Reference	374
6.53	Transform4.cc File Reference	375
6.54	Transform4.hh File Reference	376
6.55	Transform8.cc File Reference	377
6.56	Transform8.hh File Reference	378
6.57	TransformS.cc File Reference	379
6.58	TransformS.hh File Reference	380
6.59	X86-64.c File Reference	381
6.60	X86-64.h File Reference	382
6.61	X86.c File Reference	383
6.62	X86.h File Reference	384

Chapter 1

libDSP Main Page

DSP operations

Copyright (C) 1998-2003 Jussi Laako

Author:

Jussi Laako

Date

2004/02/29 17:31:23

Overview

This set of classes implement routines commonly used in digital signal processing.

Licensing

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

Chapter 2

libDSP Hierarchical Index

2.1 libDSP Class Hierarchy

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

_sDCplx	9
_sDPolar	10
_sSCplx	11
_sSPolar	12
_uDCoord	13
_uSCoord	14
clAlloc	15
cIDSPAlloc	24
clCondition	22
clDSPOp	25
clFilter	133
clDynThreads< TThreads >::_stParams2	120
clDynThreadsBase	121
clDynThreads< TThreads >	118
clDynThreadsBase::_stParams	123
clException	124
clFFTMultiRate	130
clFFTDecimator	126
clFFTInterpolator	128
clFIRMultiRate	148
clFIRDecimator	144
clFIRInterpolator	146
clFlipBand	151
clHankel	154
clIIRCascade	159
clIIRMultiRate	166
clIIRDecimator	162
clIIRInterpolator	164

clMutex	168
clPthCond	170
clPthMutex	171
clReBuffer	172
clReBufferT< TReBuffer_t >	175
clReBufferT< TDSPVector_t >	175
clDSPVector< TDSPVector_t >	111
clRecDecimator	180
clRecInterpolator	185
clRWLock	190
clSemaphore	193
clTransform4	196
clTransform8	205
clTransformS	209
clUserThreads	213

Chapter 3

libDSP Class Index

3.1 libDSP Class List

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

_sDCplx (Double precision complex datatype)	9
_sDPolar (Double precision polar datatype)	10
_sSCplx (Single precision complex datatype)	11
_sSPolar (Single precision polar datatype)	12
_uDCoord (Double precision cartesian/polar datatype)	13
_uSCoord (Single precision cartesian/polar datatype)	14
clAlloc (Class for memory allocation operations)	15
clCondition (Class implementation of POSIX condition variable)	22
clDSPAlloc (Class specialization to support automatic typecasts to cartesian/polar datatypes)	24
clDSPOp (Class of DSP operations)	25
clDSPVector< TDSPVector_t >	111
clDynThreads< TThreads >	118
clDynThreads< TThreads >::_stParams2	120
clDynThreadsBase	121
clDynThreadsBase::_stParams	123
clException	124
clFFTDecimator (FFT decimation filter class implementation)	126
clFFTInterpolator (FFT interpolation filter class implementation)	128
clFFTMultiRate (Base class for FFT based multirate filters)	130
clFilter (Class implementing FFT-based FIR-filter with design functions)	133
clFIRDecimator (FIR decimation filter class implementation)	144
clFIRInterpolator (FIR interpolation filter class implementation)	146
clFIRMultiRate (Base class for FIR based multirate filters)	148
clFlipBand	151
clHankel (Class implementation of (modified) Hankel-transform)	154
clIIRCascade (Class to handle cascaded IIR stages as one filter)	159
clIIRDecimator (IIR decimation filter class implementation)	162
clIIRInterpolator (IIR interpolation filter class implementation)	164

clIIRMultiRate (Base class for IIR based multirate filters)	166
clMutex (Class implementation of POSIX mutex semaphore)	168
clPthCond	170
clPthMutex	171
clReBuffer (Class for splitting data into buffers of different sizes)	172
clReBufferT< TReBuffer_t > (Template class for splitting data into buffers of different sizes)	175
clRecDecimator (Recursive decimation filter class implementation)	180
clRecInterpolator (Recursive interpolation filter class implementation)	185
clRWLock (Class implementation of SUSv2 read-write locks)	190
clSemaphore (Class implementation of POSIX counting semaphores)	193
clTransform4 (Decimation-in-frequency radix-2/4 transform)	196
clTransform8 (Decimation-in-frequency radix-2/4/8 transform)	205
clTransformS (Decimation-in-frequency split-radix transform)	209
clUserThreads	213

Chapter 4

libDSP File Index

4.1 libDSP File List

Here is a list of all files with brief descriptions:

Alloc.hh	215
cdspop.cc	216
Compilers.hh	274
Condition.hh	275
DSPConfig.hh	276
DSPOp.cc	277
dspop.h	278
DSPOp.hh	327
dsptypes.h	328
DSPVector.hh	331
DynThreads.cc	332
DynThreads.hh	333
Exception.hh	334
FFTDecimator.cc	335
FFTDecimator.hh	336
FFTInterpolator.cc	337
FFTInterpolator.hh	338
FFTMultiRate.cc	339
FFTMultiRate.hh	340
Filter.cc	341
Filter.hh	342
FIRDecimator.cc	343
FIRDecimator.hh	344
FIRInterpolator.cc	345
FIRInterpolator.hh	346
FIRMultiRate.cc	347
FIRMultiRate.hh	348
FlipBand.cc	349
FlipBand.hh	350

Hankel.cc	351
Hankel.hh	353
IIRCascade.cc	354
IIRCascade.hh	355
IIRDecimator.cc	356
IIRDecimator.hh	357
IIRInterpolator.cc	358
IIRInterpolator.hh	359
IIRMultiRate.cc	360
IIRMultiRate.hh	361
Mutex.hh	362
PthCond.hh	363
PthMutex.hh	364
ReBuffer.cc	365
ReBuffer.hh	366
ReBufferT.hh	367
RecDecimator.cc	368
RecDecimator.hh	369
RecInterpolator.cc	370
RecInterpolator.hh	371
RWLock.hh	372
Semaphore.hh	373
Test.cc	374
Transform4.cc	375
Transform4.hh	376
Transform8.cc	377
Transform8.hh	378
TransformS.cc	379
TransformS.hh	380
X86-64.c	381
X86-64.h	382
X86.c	383
X86.h	384

Chapter 5

libDSP Class Documentation

5.1 `_sDCplx` Struct Reference

Double precision complex datatype.

```
#include <dsptypes.h>
```

Public Attributes

- double [R](#)
Real.
- double [I](#)
Imaginary.

5.1.1 Detailed Description

Double precision complex datatype.

5.1.2 Member Data Documentation

5.1.2.1 `double _sDCplx::R`

Real.

5.1.2.2 `double _sDCplx::I`

Imaginary.

5.2 `_sDPolar` Struct Reference

Double precision polar datatype.

```
#include <dsptypes.h>
```

Public Attributes

- double `M`
Magnitude.
- double `P`
Phase.

5.2.1 Detailed Description

Double precision polar datatype.

5.2.2 Member Data Documentation

5.2.2.1 double `_sDPolar::M`

Magnitude.

5.2.2.2 double `_sDPolar::P`

Phase.

5.3 _sSCplx Struct Reference

Single precision complex datatype.

```
#include <dsptypes.h>
```

Public Attributes

- float [R](#)
Real.
- float [I](#)
Imaginary.

5.3.1 Detailed Description

Single precision complex datatype.

Type prefix is left out to prevent need of source code changes when precision is changed.

5.3.2 Member Data Documentation

5.3.2.1 float [_sSCplx::R](#)

Real.

5.3.2.2 float [_sSCplx::I](#)

Imaginary.

5.4 `_sSPolar` Struct Reference

Single precision polar datatype.

```
#include <dsptypes.h>
```

Public Attributes

- float [M](#)
Magnitude.
- float [P](#)
Phase.

5.4.1 Detailed Description

Single precision polar datatype.

5.4.2 Member Data Documentation

5.4.2.1 float [_sSPolar::M](#)

Magnitude.

5.4.2.2 float [_sSPolar::P](#)

Phase.

5.5 `_uDCoord` Union Reference

Double precision cartesian/polar datatype.

```
#include <dsptypes.h>
```

Public Attributes

- `stDCplx C`
Cartesian.
- `stDPolar P`
Polar.

5.5.1 Detailed Description

Double precision cartesian/polar datatype.

5.5.2 Member Data Documentation

5.5.2.1 `stDCplx _uDCoord::C`

Cartesian.

5.5.2.2 `stDPolar _uDCoord::P`

Polar.

5.6 `_uSCoord` Union Reference

Single precision cartesian/polar datatype.

```
#include <dsptypes.h>
```

Public Attributes

- [stSCplx C](#)
Cartesian.
- [stSPolar P](#)
Polar.

5.6.1 Detailed Description

Single precision cartesian/polar datatype.

5.6.2 Member Data Documentation

5.6.2.1 [stSCplx _uSCoord::C](#)

Cartesian.

5.6.2.2 [stSPolar _uSCoord::P](#)

Polar.

5.7 clAlloc Class Reference

Class for memory allocation operations.

#include <Alloc.hh>

Inherited by [clDSPAlloc](#).

Public Member Functions

- [clAlloc](#) ()
Constructor; initializes empty allocation.
- [clAlloc](#) (const [clAlloc](#) &CopySrc)
Copy constructor; copies instance.
- [clAlloc](#) (long lAllocSize)
Constructor; allocates specified amount of memory.
- [~clAlloc](#) ()
Destructor; frees allocated memory block.
- void * [Size](#) (long lAllocSize)
Allocate specified amount of memory, previously allocated block is freed before allocating new one.
- void * [Resize](#) (long lAllocSize)
Resize memory block to specified size.
- void [Free](#) ()
Free allocated memory block.
- void * [GetPtr](#) () const
Get pointer to allocated block.
- long [GetSize](#) () const
Get size of allocation.
- void [Lock](#) ()
Lock memory block (prevent paging out).
- void [UnLock](#) ()
Unlock memory block (allow paging out).
- void [Copy](#) (const [clAlloc](#) &Src)
Copy specified memory block to this instance.

- void `CopyTo (clAlloc &Dest)`
Copy contents of this instance to specified memory block.
- `clAlloc GetCopy ()`
Return copy of this instance.
- `operator char * () const`
Return pointer to memory block.
- `operator unsigned char * () const`
- `operator short * () const`
- `operator unsigned short * () const`
- `operator int * () const`
- `operator unsigned int * () const`
- `operator long * () const`
- `operator unsigned long * () const`
- `operator float * () const`
- `operator double * () const`
- `operator long double * () const`
- `operator void * () const`
- `operator void * ()`
- `clAlloc & operator= (const clAlloc &Src)`
Self explanatory.

Protected Attributes

- bool `bLocked`
- long `lSize`
- void * `vpPtr`

5.7.1 Detailed Description

Class for memory allocation operations.

Mainly to avoid memory leaks and to simplify pointer typecasts.

5.7.2 Constructor & Destructor Documentation

5.7.2.1 `clAlloc::clAlloc () [inline]`

Constructor; initializes empty allocation.

5.7.2.2 `clAlloc::clAlloc (const clAlloc & CopySrc) [inline]`

Copy constructor; copies instance.

5.7.2.3 clAlloc::clAlloc (long lAllocSize) [inline]

Constructor; allocates specified amount of memory.

Parameters:

lAllocSize Size, in bytes, of allocation

Exceptions:

runtime_error

5.7.2.4 clAlloc::~clAlloc () [inline]

Destructor; frees allocated memory block.

5.7.3 Member Function Documentation

5.7.3.1 void* clAlloc::Size (long lAllocSize) [inline]

Allocate specified amount of memory, previously allocated block is freed before allocating new one.

Parameters:

lAllocSize Size, in bytes, of new allocation

Returns:

Pointer to memory block

Exceptions:

runtime_error

5.7.3.2 void* clAlloc::Resize (long lAllocSize) [inline]

Resize memory block to specified size.

Parameters:

lAllocSize New size, in bytes, of memory block

Returns:

Pointer to memory block

Exceptions:

runtime_error

5.7.3.3 void clAlloc::Free () [inline]

Free allocated memory block.

It is not an error to free already freed or non-allocated block.

5.7.3.4 void* clAlloc::GetPtr () const [inline]

Get pointer to allocated block.

Returns:

Pointer to memory block

5.7.3.5 long clAlloc::GetSize () const [inline]

Get size of allocation.

Returns:

Size, in bytes, of allocation

5.7.3.6 void clAlloc::Lock () [inline]

Lock memory block (prevent paging out).

5.7.3.7 void clAlloc::UnLock () [inline]

Unlock memory block (allow paging out).

5.7.3.8 void clAlloc::Copy (const clAlloc & Src) [inline]

Copy specified memory block to this instance.

Parameters:

Src Source of copy

Exceptions:

runtime_error

5.7.3.9 void clAlloc::CopyTo (clAlloc & Dest) [inline]

Copy contents of this instance to specified memory block.

Parameters:

Dest Destination of copy

Exceptions:*runtime_error***5.7.3.10 clAlloc clAlloc::GetCopy () [inline]**

Return copy of this instance.

Returns:

Copy of this memory block

5.7.3.11 clAlloc::operator char * () const [inline]

Return pointer to memory block.

5.7.3.12 clAlloc::operator unsigned char * () const [inline]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.7.3.13 clAlloc::operator short * () const [inline]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.7.3.14 clAlloc::operator unsigned short * () const [inline]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.7.3.15 clAlloc::operator int * () const [inline]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.7.3.16 clAlloc::operator unsigned int * () const [inline]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.7.3.17 clAlloc::operator long * () const [inline]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.7.3.18 clAlloc::operator unsigned long * () const [inline]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.7.3.19 clAlloc::operator float * () const [inline]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.7.3.20 clAlloc::operator double * () const [inline]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.7.3.21 clAlloc::operator long double * () const [inline]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.7.3.22 clAlloc::operator void * () const [inline]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.7.3.23 clAlloc::operator void * () [inline]**5.7.3.24 clAlloc& clAlloc::operator= (const clAlloc & Src)** [inline]

Self explanatory.

Parameters:

Src Source of copy

Returns:

Reference to this

Exceptions:

runtime_error

5.7.4 Member Data Documentation

5.7.4.1 **bool** [clAlloc::bLocked](#) [protected]

5.7.4.2 **long** [clAlloc::lSize](#) [protected]

5.7.4.3 **void*** [clAlloc::vpPtr](#) [protected]

5.8 clCondition Class Reference

Class implementation of POSIX condition variable.

```
#include <Condition.hh>
```

Public Member Functions

- [clCondition](#) ()
Constructor; creates and initializes the condition variable.
- [~clCondition](#) ()
Destructor; destroys the condition variable.
- bool [Wait](#) (pthread_mutex_t *pThmCond)
Wait for condition to be signalled.
- bool [Wait](#) (pthread_mutex_t *pThmCond, int iCondTO)
- void [Notify](#) ()
Signal condition variable.
- void [NotifyAll](#) ()
Signal condition variable.

Private Attributes

- pthread_cond_t [pthcCond](#)

5.8.1 Detailed Description

Class implementation of POSIX condition variable.

5.8.2 Constructor & Destructor Documentation

5.8.2.1 clCondition::clCondition () [inline]

Constructor; creates and initializes the condition variable.

5.8.2.2 clCondition::~~clCondition () [inline]

Destructor; destroys the condition variable.

5.8.3 Member Function Documentation

5.8.3.1 `bool clCondition::Wait (pthread_mutex_t * pthmCond)` [inline]

Wait for condition to be signalled.

Note:

Mutex must be locked before calling this. Mutex is freed for the waiting time and relocked on return.

Parameters:

pthmCond Pointer to mutex variable

Returns:

Success

5.8.3.2 `bool clCondition::Wait (pthread_mutex_t * pthmCond, int iCondTO)` [inline]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Parameters:

pthmCond Pointer to mutex variable

iCondTO Timeout in milliseconds

Returns:

Success

5.8.3.3 `void clCondition::Notify ()` [inline]

Signal condition variable.

Mutex can be locked to achieve desired scheduling behaviour. Only single waiting thread is woken up.

5.8.3.4 `void clCondition::NotifyAll ()` [inline]

Signal condition variable.

Mutex can be locked to achieve desired scheduling behaviour. All waiting threads are woken up.

5.8.4 Member Data Documentation

5.8.4.1 `pthread_cond_t clCondition::pthcCond` [private]

5.9 cDSPAlloc Class Reference

Class specialization to support automatic typecasts to cartesian/polar datatypes.

```
#include <DSPOp.hh>
```

Inherits [cAlloc](#).

Public Member Functions

- [cDSPAlloc](#) ()
- [cDSPAlloc](#) (const [cDSPAlloc](#) &CopySrc)
- [cDSPAlloc](#) (long lAllocSize)
- [operator stSCplx *](#) ()
- [operator stDCplx *](#) ()
- [operator stSPolar *](#) ()
- [operator stDPolar *](#) ()

5.9.1 Detailed Description

Class specialization to support automatic typecasts to cartesian/polar datatypes.

5.9.2 Constructor & Destructor Documentation

5.9.2.1 [cDSPAlloc::cDSPAlloc](#) () [inline]

5.9.2.2 [cDSPAlloc::cDSPAlloc](#) (const [cDSPAlloc](#) & *CopySrc*) [inline]

5.9.2.3 [cDSPAlloc::cDSPAlloc](#) (long *lAllocSize*) [inline]

5.9.3 Member Function Documentation

5.9.3.1 [cDSPAlloc::operator stSCplx *](#) () [inline]

5.9.3.2 [cDSPAlloc::operator stDCplx *](#) () [inline]

5.9.3.3 [cDSPAlloc::operator stSPolar *](#) () [inline]

5.9.3.4 [cDSPAlloc::operator stDPolar *](#) () [inline]

5.10 clDSPOp Class Reference

Class of DSP operations.

#include <DSPOp.hh>

Inherited by [clFilter](#)[private].

Public Member Functions

- [clDSPOp](#) ()
- [~clDSPOp](#) ()
- void [WinBartlett](#) (float *, long)
Calculate Bartlett (triangle) window coefficients.
- void [WinBartlett](#) (double *, long)
- void [WinBlackman](#) (float *, long)
Calculate Blackman window coefficients.
- void [WinBlackman](#) (double *, long)
- void [WinBlackman](#) (float *, long, float)
Calculate blackman window coefficients with specified alpha.
- void [WinBlackman](#) (double *, long, double)
- void [WinBlackmanHarris](#) (float *, long)
Calculate Blackman-Harris window coefficients.
- void [WinBlackmanHarris](#) (double *, long)
- void [WinCosTapered](#) (float *, long)
Calculate cosine tapered window coefficients.
- void [WinCosTapered](#) (double *, long)
- void [WinCosTaperedA](#) (float *, long)
- void [WinCosTaperedA](#) (double *, long)
- void [WinCosTaperedA](#) (float *, const float *, long)
- void [WinCosTaperedA](#) (double *, const double *, long)
- void [WinExactBlackman](#) (float *, long)
Calculate exact Blackman window coefficients.
- void [WinExactBlackman](#) (double *, long)
- void [WinExp](#) (float *, float, long)
Calculate exponential window coefficients.
- void [WinExp](#) (double *, double, long)
- void [WinFlatTop](#) (float *, long)
Calculate flat top window coefficients.

- void [WinFlatTop](#) (double *, long)
- void [WinGenericCos](#) (float *, long, const float *, long)
Calculate generic cosine window coefficients.
- void [WinGenericCos](#) (double *, long, const double *, long)
- void [WinHamming](#) (float *, long)
Calculate Hamming window coefficients.
- void [WinHamming](#) (double *, long)
- void [WinHanning](#) (float *, long)
Calculate Hanning (Hann?) window coefficients.
- void [WinHanning](#) (double *, long)
- void [WinKaiser](#) (float *, float, long)
Calculate Kaiser window coefficients See ::ModZeroBessel() for I_0 .
- void [WinKaiser](#) (double *, double, long)
- void [WinKaiserBessel](#) (float *, float, long)
Calculate Kaiser-Bessel window coefficients See ::ModZeroBessel() for I_0 .
- void [WinKaiserBessel](#) (double *, double, long)
- void [WinTukey](#) (float *, long)
Calculate Tukey window coefficients.
- void [WinTukey](#) (double *, long)
- void [WinDolphChebyshev](#) (float *, float, long)
Calculate Dolph-Chebyshev window coefficients.
- void [WinDolphChebyshev](#) (double *, double, long)
- void [Spatialize](#) (float *, float *, const float *, long)
Spatialize one channel to two channels by inverting others phase.
- void [Spatialize](#) (double *, double *, const double *, long)
- void [Spatialize](#) (float *, const float *, long)
- void [Spatialize](#) (double *, const double *, long)
- long [ReBuffer](#) (float *, const float *, long, long)
Rebuffer to different sized buffer.
- long [ReBuffer](#) (double *, const double *, long, long)
- double [DegToRad](#) (double dSource)
Convert degrees to radians.
- float [DegToRad](#) (float fSource)
- double [RadToDeg](#) (double dSource)
Convert radians to degrees.

- float [RadToDeg](#) (float fSource)
- void [FIRAllocate](#) (const float *, long)
Allocate FIR filter.
- void [FIRAllocate](#) (const double *, long)
- void [FIRFilter](#) (float *, long)
Filter using FIR filter.
- void [FIRFilter](#) (double *, long)
- void [FIRFilter](#) (float *, const float *, long)
- void [FIRFilter](#) (double *, const double *, long)
- void [FIRFilterF](#) (float *, float *, long)
Filter using FIR filter (fast version).
- void [FIRFilterF](#) (double *, double *, long)
- void [FIRFree](#) ()
Free FIR filter.
- void [IIRInitialize](#) (const float *)
Initialize IIR filter.
- void [IIRInitialize](#) (const double *)
- void [IIRFilter](#) (float *, long)
Filter using IIR filter.
- void [IIRFilter](#) (double *, long)
- void [IIRFilter](#) (float *, const float *, long)
- void [IIRFilter](#) (double *, const double *, long)
- void [IIRClear](#) ()
Clear IIR filter feedback chain.
- void [FFTInitialize](#) (long, bool)
Initialize FFT (and other transforms).
- void [FFTUninitialize](#) ()
Uninitialize FFT.
- void [FFTi](#) (stpSCplx, float *)
FFT half in-place, source vector is modified.
- void [FFTi](#) (stpDCplx, double *)
- void [FFTTo](#) (stpSCplx, const float *)
FFT out-of-place.
- void [FFTTo](#) (stpDCplx, const double *)
- void [FFTTo](#) (stpSCplx, const stpSCplx)

- void [FFTo](#) ([stpDCplx](#), const [stpDCplx](#))
- void [IFFTo](#) (float *, const [stpSCplx](#))
IFFT out-of-place.
- void [IFFTo](#) (double *, const [stpDCplx](#))
- void [IFFTo](#) ([stpSCplx](#), const [stpSCplx](#))
- void [IFFTo](#) ([stpDCplx](#), const [stpDCplx](#))

Static Public Member Functions

- signed long [Round](#) (float)
Round floatingpoint number to integer.
- signed long [Round](#) (double)
- void [Add](#) (float *, float, long)
Add single value to vector.
- void [Add](#) (double *, double, long)
- void [Add](#) ([stpSCplx](#), [stSCplx](#), long)
- void [Add](#) ([stpDCplx](#), [stDCplx](#), long)
- void [Add](#) (float *, const float *, long)
Add two vectors.
- void [Add](#) (double *, const double *, long)
- void [Add](#) ([stpSCplx](#), const [stpSCplx](#), long)
- void [Add](#) ([stpDCplx](#), const [stpDCplx](#), long)
- void [Add](#) (float *, const float *, const float *, long)
Add two vectors and return result in third.
- void [Add](#) (double *, const double *, const double *, long)
- void [Add](#) ([stpSCplx](#), const [stpSCplx](#), const [stpSCplx](#), long)
- void [Add](#) ([stpDCplx](#), const [stpDCplx](#), const [stpDCplx](#), long)
- void [Sub](#) (float *, float, long)
Subtract single value from vector.
- void [Sub](#) (double *, double, long)
- void [Sub](#) ([stpSCplx](#), [stSCplx](#), long)
- void [Sub](#) ([stpDCplx](#), [stDCplx](#), long)
- void [Sub](#) (float *, const float *, long)
Subtract two vectors.
- void [Sub](#) (double *, const double *, long)
- void [Sub](#) ([stpSCplx](#), const [stpSCplx](#), long)
- void [Sub](#) ([stpDCplx](#), const [stpDCplx](#), long)
- void [Sub](#) (float *, const float *, const float *, long)
Subtract two vectors and return result in third.

- void [Sub](#) (double *, const double *, const double *, long)
- void [Sub](#) ([stpSCplx](#), const [stpSCplx](#), const [stpSCplx](#), long)
- void [Sub](#) ([stpDCplx](#), const [stpDCplx](#), const [stpDCplx](#), long)
- void [Mul](#) (float *, float, long)

Multiply vector with single value in-place.

- void [Mul](#) (double *, double, long)
- void [Mul](#) ([stpSCplx](#), float, long)
- void [Mul](#) ([stpDCplx](#), double, long)
- void [Mul](#) ([stpSCplx](#), [stSCplx](#), long)
- void [Mul](#) ([stpDCplx](#), [stDCplx](#), long)
- void [Mul](#) (float *, const float *, float, long)

Multiply vector with single value out-of-place.

- void [Mul](#) (double *, const double *, double, long)
- void [Mul](#) (float *, const float *, long)

Multiply two vectors.

- void [Mul](#) (double *, const double *, long)
- void [Mul](#) ([stpSCplx](#), const float *, long)
- void [Mul](#) ([stpDCplx](#), const double *, long)
- void [Mul](#) ([stpSCplx](#), const [stpSCplx](#), long)
- void [Mul](#) ([stpDCplx](#), const [stpDCplx](#), long)
- void [Mul](#) (float *, const float *, const float *, long)

Multiply two vectors and return result in third.

- void [Mul](#) (double *, const double *, const double *, long)
- void [Mul](#) ([stpSCplx](#), const [stpSCplx](#), const [stpSCplx](#), long)
- void [Mul](#) ([stpDCplx](#), const [stpDCplx](#), const [stpDCplx](#), long)
- void [MulC](#) ([stpSCplx](#), const [stpSCplx](#), long)

Multiply vector with others complex conjugate.

- void [MulC](#) ([stpDCplx](#), const [stpDCplx](#), long)
- void [MulC](#) ([stpSCplx](#), const [stpSCplx](#), const [stpSCplx](#), long)
- void [MulC](#) ([stpDCplx](#), const [stpDCplx](#), const [stpDCplx](#), long)
- void [Mul2](#) (float *, float *, const float *, long)

Multiply two vectors in-place with third vector.

- void [Mul2](#) (double *, double *, const double *, long)
- void [Mul2](#) (float *, float *, const float *, const float *, const float *, long)

Multiply two vectors out-of-place with third vector and return results in separate two vectors.

- void [Mul2](#) (double *, double *, const double *, const double *, const double *, long)

- void [Div](#) (float *, float, long)

Divide vector with single value.

- void [Div](#) (double *, double, long)
- void [Div](#) ([stpSCplx](#), [stSCplx](#), long)
- void [Div](#) ([stpDCplx](#), [stDCplx](#), long)
- void [Div](#) (float *, const float *, long)

Divide two vectors.

- void [Div](#) (double *, const double *, long)
- void [Div](#) ([stpSCplx](#), const [stpSCplx](#), long)
- void [Div](#) ([stpDCplx](#), const [stpDCplx](#), long)
- void [Div](#) (float *, const float *, const float *, long)

Divide two vectors and return result in third.

- void [Div](#) (double *, const double *, const double *, long)
- void [Div](#) ([stpSCplx](#), const [stpSCplx](#), const [stpSCplx](#), long)
- void [Div](#) ([stpDCplx](#), const [stpDCplx](#), const [stpDCplx](#), long)
- void [Div1x](#) (float *, long)

Calculate 1 / value in-place.

- void [Div1x](#) (double *, long)
- void [Div1x](#) (float *, const float *, long)

Calculate 1 / value out-of-place.

- void [Div1x](#) (double *, const double *, long)
- void [MulAdd](#) (float *, float, float, long)

Multiply-add vector in-place.

- void [MulAdd](#) (double *, double, double, long)
- void [MulAdd](#) (float *, const float *, float, float, long)

Multiply-add vector out-of-place.

- void [MulAdd](#) (double *, const double *, double, double, long)
- void [Abs](#) (float *, long)

Get absolute value in-place.

- void [Abs](#) (double *, long)
- void [Abs](#) (float *, const float *, long)

Get absolute value out-of-place.

- void [Abs](#) (double *, const double *, long)
- void [Sqrt](#) (float *, long)

Calculate square-roots in-place.

- void [Sqrt](#) (double *, long)

- void [Sqrt](#) (float *, const float *, long)
Calculate square-roots out-of-place.
- void [Sqrt](#) (double *, const double *, long)
- void [Zero](#) (float *, long)
Set all vector elements to zero.
- void [Zero](#) (double *, long)
- void [Zero](#) (stpSCplx, long)
- void [Zero](#) (stpDCplx, long)
- void [Set](#) (float *, float, long)
Set all vector elements to specified value.
- void [Set](#) (double *, double, long)
- void [Set](#) (stpSCplx, stSCplx, long)
- void [Set](#) (stpDCplx, stDCplx, long)
- void [Set](#) (float *, float, long, long, long)
Set specified vector elements to specified value with boundary check.
- void [Set](#) (double *, double, long, long, long)
- void [Set](#) (stpSCplx, stSCplx, long, long, long)
- void [Set](#) (stpDCplx, stDCplx, long, long, long)
- void [Clip](#) (float *, float, long)
Clip vector in-place to specified value.
- void [Clip](#) (double *, double, long)
- void [Clip](#) (float *, const float *, float, long)
Clip vector out-of-place to specified value.
- void [Clip](#) (double *, const double *, double, long)
- void [Clip](#) (float *, float, float, long)
Clip vector in-place to fit between specified values.
- void [Clip](#) (double *, double, double, long)
- void [Clip](#) (float *, const float *, float, float, long)
Clip vector out-of-place to fit between specified values.
- void [Clip](#) (double *, const double *, double, double, long)
- void [ClipZero](#) (float *, long)
Clip smallest values in vector to zero (in-place).
- void [ClipZero](#) (double *, long)
- void [ClipZero](#) (float *, const float *, long)
Clip smallest values in vector to zero (out-of-place).
- void [ClipZero](#) (double *, const double *, long)

- void [Copy](#) (float *, const float *, long)
Copy source vector to destination vector(s).
- void [Copy](#) (double *, const double *, long)
- void [Copy](#) (stpSCplx, const stpSCplx, long)
- void [Copy](#) (stpDCplx, const stpDCplx, long)
- void [Copy](#) (float *, float *, const float *, long)
- void [Copy](#) (double *, double *, const double *, long)
- float [Convolve](#) (const float *, const float *, long)
Convolve two vectors (same length, finite) without lag.
- double [Convolve](#) (const double *, const double *, long)
- void [Convolve](#) (float *, const float *, const float *, long)
Convolve two vectors (same length, finite) with lag.
- void [Convolve](#) (double *, const double *, const double *, long)
- float [Correlate](#) (const float *, const float *, long)
Correlate two vectors (same length, finite) without lag.
- double [Correlate](#) (const double *, const double *, long)
- void [Correlate](#) (float *, const float *, const float *, long)
Correlate two vectors (same length, finite) with lag.
- void [Correlate](#) (double *, const double *, const double *, long)
- float [AutoCorrelate](#) (const float *, long)
Autocorrelate vector.
- double [AutoCorrelate](#) (const double *, long)
- void [AutoCorrelate](#) (float *, const float *, long)
Autocorrelate vector.
- void [AutoCorrelate](#) (double *, const double *, long)
- float [DotProduct](#) (const float *, const float *, long)
Dot product of two vectors.
- double [DotProduct](#) (const double *, const double *, long)
- void [MinMax](#) (float *, float *, const float *, long)
Find minimum and maximum values of vector.
- void [MinMax](#) (double *, double *, const double *, long)
- float [Mean](#) (const float *, long)
Calculate mean of vector.
- double [Mean](#) (const double *, long)
- float [Median](#) (const float *, long)
Calculate median of vector.

- double [Median](#) (const double *, long)
- void [Negate](#) (float *, long)
Negate vector elements.
- void [Negate](#) (double *, long)
- void [Negate](#) (float *, const float *, long)
Negate vector elements.
- void [Negate](#) (double *, const double *, long)
- void [Normalize](#) (float *, long)
Normalize vector elements.
- void [Normalize](#) (double *, long)
- void [Normalize](#) (float *, const float *, long)
Normalize vector elements.
- void [Normalize](#) (double *, const double *, long)
- float [Product](#) (const float *, long)
Product of vector elements.
- double [Product](#) (const double *, long)
- void [Reverse](#) (float *, long)
Reverse vector (in-place).
- void [Reverse](#) (double *, long)
- void [Reverse](#) (stpSCplx, long)
- void [Reverse](#) (stpDCplx, long)
- void [Reverse](#) (float *, const float *, long)
Reverse vector (out-of-place).
- void [Reverse](#) (double *, const double *, long)
- void [Reverse](#) (stpSCplx, const stpSCplx, long)
- void [Reverse](#) (stpDCplx, const stpDCplx, long)
- void [Scale](#) (float *, long)
Scale (normalize) vector to range [-1:1].
- void [Scale](#) (double *, long)
- void [Scale](#) (float *, const float *, long)
- void [Scale](#) (double *, const double *, long)
- void [Scale01](#) (float *, long)
Scale (normalize) vector to range [0:1].
- void [Scale01](#) (double *, long)
- void [Scale01](#) (float *, const float *, long)
- void [Scale01](#) (double *, const double *, long)

- void [Sort](#) (float *, long)
Sort vector elements (in-place).
- void [Sort](#) (double *, long)
- void [Sort](#) (long *, long)
- void [StdDev](#) (float *, float *, const float *, long)
Calculate standard deviation and mean of vector.
- void [StdDev](#) (double *, double *, const double *, long)
- float [Sum](#) (const float *, long)
Calculate sum of vector elements.
- double [Sum](#) (const double *, long)
- void [Square](#) (float *, long)
Square vector in-place.
- void [Square](#) (double *, long)
- void [Square](#) (float *, const float *, long)
Square vector out-of-place.
- void [Square](#) (double *, const double *, long)
- void [Convert](#) (float *, const unsigned char *, long)
Vector datatype conversion.
- void [Convert](#) (float *, const signed short *, long, bool)
- void [Convert](#) (float *, const signed int *, long, bool)
- void [Convert](#) (float *, const double *, long)
- void [Convert](#) (double *, const unsigned char *, long)
- void [Convert](#) (double *, const signed short *, long, bool)
- void [Convert](#) (double *, const signed int *, long, bool)
- void [Convert](#) (double *, const float *, long)
- void [Convert](#) (unsigned char *, const float *, long)
- void [Convert](#) (unsigned char *, const double *, long)
- void [Convert](#) (signed short *, const float *, long, bool)
- void [Convert](#) (signed short *, const double *, long, bool)
- void [Convert](#) (signed int *, const float *, long, bool)
- void [Convert](#) (signed int *, const double *, long, bool)
- void [CartToPolar](#) (float *, float *, const float *, const float *, long)
Convert cartesian to polar vectors (out-of-place).
- void [CartToPolar](#) (double *, double *, const double *, const double *, long)
- void [CartToPolar](#) (float *, float *, const [stpSCplx](#), long)
- void [CartToPolar](#) (double *, double *, const [stpDCplx](#), long)
- void [CartToPolar](#) ([stpSPolar](#), const [stpSCplx](#), long)
- void [CartToPolar](#) ([stpDPolar](#), const [stpDCplx](#), long)
- void [CartToPolar](#) ([utpSCoord](#), long)

Convert cartesian to polar vectors (in-place).

- void [CartToPolar](#) (utpDCoord, long)
- void [PolarToCart](#) (float *, float *, const float *, const float *, long)

Convert polar to cartesian vectors (out-of-place).

- void [PolarToCart](#) (double *, double *, const double *, const double *, long)
- void [PolarToCart](#) (stpSCplx, const float *, const float *, long)
- void [PolarToCart](#) (stpDCplx, const double *, const double *, long)
- void [PolarToCart](#) (stpSCplx, const stpSPolar, long)
- void [PolarToCart](#) (stpDCplx, const stpDPolar, long)
- void [PolarToCart](#) (utpSCoord, long)

Convert cartesian to polar vectors (in-place).

- void [PolarToCart](#) (utpDCoord, long)
- float [CrossCorr](#) (const float *, const float *, long)

Calculate normalized cross correlation of two vectors without delay.

- double [CrossCorr](#) (const double *, const double *, long)
- float [DelCrossCorr](#) (const float *, const float *, long, long)

Calculate normalized cross correlation of two vectors with delay in second.

- double [DelCrossCorr](#) (const double *, const double *, long, long)
- void [DelCrossCorr](#) (float *, const float *, const float *, long, const long *, long)

Calculate normalized cross correlation of two vectors with delays specified in vector.

- void [DelCrossCorr](#) (double *, const double *, const double *, long, const long *, long)
- float [Energy](#) (const float *, long)

Calculates energy (square) of vector.

- double [Energy](#) (const double *, long)
- void [Magnitude](#) (float *, const stpSCplx, long)

Calculates magnitudes (linear) of vector.

- void [Magnitude](#) (double *, const stpDCplx, long)
- void [Power](#) (float *, const stpSCplx, long)

Calculates powers (in dB) of vector.

- void [Power](#) (double *, const stpDCplx, long)
- void [Phase](#) (float *, const stpSCplx, long)

Calculated phases (in rad) of vector.

- void [Phase](#) (double *, const stpDCplx, long)
- void [PowerPhase](#) (float *, float *, const stpSCplx, long)

Calculates powers (in dB) and phases (in rad) of vector See ::Power() and ::Phase() for formulas.

- void [PowerPhase](#) (double *, double *, const [stpDCplx](#), long)
- void [Decimate](#) (float *, const float *, long, long)

Decimate vector without average.

- void [Decimate](#) (double *, const double *, long, long)
- void [DecimateAvg](#) (float *, const float *, long, long)

Decimate vector with average.

- void [DecimateAvg](#) (double *, const double *, long, long)
- void [Interpolate](#) (float *, const float *, long, long)

Interpolate vector without average (nulling).

- void [Interpolate](#) (double *, const double *, long, long)
- void [InterpolateAvg](#) (float *, const float *, long, long)

Interpolate vector with average (linear).

- void [InterpolateAvg](#) (double *, const double *, long, long)
- void [Resample](#) (float *, long, const float *, long)

Resample vector to different length.

- void [Resample](#) (double *, long, const double *, long)
- void [ResampleAvg](#) (float *, long, const float *, long)

Resample vector to different length with average.

- void [ResampleAvg](#) (double *, long, const double *, long)
- float [RMS](#) (const float *, long)

Calculate RMS (root mean square) of vector.

- double [RMS](#) (const double *, long)
- float [Variance](#) (float *, float *, const float *, long)

Calculate variance and mean of vector.

- double [Variance](#) (double *, double *, const double *, long)
- float [PeakLevel](#) (const float *, long)

Find peak level of vector and return result in dB.

- double [PeakLevel](#) (const double *, long)
- void [Mix](#) (float *, const float *, long)

Mix two channels interleaved in vector.

- void [Mix](#) (double *, const double *, long)
- void [Mix](#) (float *, const float *, const float *, long)
- void [Mix](#) (double *, const double *, const double *, long)
- void [Mix](#) (float *, const float *, long, long)
- void [Mix](#) (double *, const double *, long, long)

- void [Extract](#) (float *, const float *, long, long, long)
Extract channel n from N channel interleaved input.
- void [Extract](#) (double *, const double *, long, long, long)
- void [Pack](#) (float *, const float *, long, long, long)
Pack channel n to N channel interleaved output.
- void [Pack](#) (double *, const double *, long, long, long)
- void [FFTWConvert](#) (stpSCplx, const float *, long)
Convert Real-FFTW complex output to our Cplx.
- void [FFTWConvert](#) (stpDCplx, const float *, long)
- void [FFTWConvert](#) (stpSCplx, const double *, long)
- void [FFTWConvert](#) (stpDCplx, const double *, long)
- void [FFTWConvert](#) (float *, const stpSCplx, long)
Convert our Cplx to Real-FFTW complex input.
- void [FFTWConvert](#) (float *, const stpDCplx, long)
- void [FFTWConvert](#) (double *, const stpSCplx, long)
- void [FFTWConvert](#) (double *, const stpDCplx, long)

Static Protected Member Functions

- void [Cart2Polar](#) (float *, float *, float, float)
Cartesian to polar conversion.
- void [Cart2Polar](#) (double *, double *, double, double)
- void [Cart2Polar](#) (float *, float *, const stpSCplx)
- void [Cart2Polar](#) (double *, double *, const stpDCplx)
- void [Cart2Polar](#) (stpSPolar, const stpSCplx)
- void [Cart2Polar](#) (stpDPolar, const stpDCplx)
- void [Cart2Polar](#) (utpSCoord)
- void [Cart2Polar](#) (utpDCoord)
- void [Polar2Cart](#) (float *, float *, float, float)
Polar to cartesian conversion.
- void [Polar2Cart](#) (double *, double *, double, double)
- void [Polar2Cart](#) (stpSCplx, float, float)
- void [Polar2Cart](#) (stpDCplx, double, double)
- void [Polar2Cart](#) (stpSCplx, const stpSPolar)
- void [Polar2Cart](#) (stpDCplx, const stpDPolar)
- void [Polar2Cart](#) (utpSCoord)
- void [Polar2Cart](#) (utpDCoord)
- void [CplxAdd](#) (stpSCplx, const stpSCplx)
Complex addition.

- void [CplxAdd](#) (stpDCplx, const stpDCplx)
- void [CplxAdd](#) (stpSCplx, const stpSCplx, const stpSCplx)
- void [CplxAdd](#) (stpDCplx, const stpDCplx, const stpDCplx)
- void [CplxSub](#) (stpSCplx, const stpSCplx)

Complex subtraction.

- void [CplxSub](#) (stpDCplx, const stpDCplx)
- void [CplxSub](#) (stpSCplx, const stpSCplx, const stpSCplx)
- void [CplxSub](#) (stpDCplx, const stpDCplx, const stpDCplx)
- void [CplxMul](#) (stpSCplx, float)

Complex multiply with real value.

- void [CplxMul](#) (stpDCplx, double)
- void [CplxMul](#) (stpSCplx, const stpSCplx)

Complex multiply.

- void [CplxMul](#) (stpDCplx, const stpDCplx)
- void [CplxMul](#) (stpSCplx, const stpSCplx, const stpSCplx)
- void [CplxMul](#) (stpDCplx, const stpDCplx, const stpDCplx)
- void [CplxMulC](#) (stpSCplx, const stpSCplx)

Complex multiply with complex conjugate.

- void [CplxMulC](#) (stpDCplx, const stpDCplx)
- void [CplxMulC](#) (stpSCplx, const stpSCplx, const stpSCplx)
- void [CplxMulC](#) (stpDCplx, const stpDCplx, const stpDCplx)
- void [CplxDiv](#) (stpSCplx, const stpSCplx)

Complex division.

- void [CplxDiv](#) (stpDCplx, const stpDCplx)
- void [CplxDiv](#) (stpSCplx, const stpSCplx, const stpSCplx)
- void [CplxDiv](#) (stpDCplx, const stpDCplx, const stpDCplx)
- void [CplxExp](#) (stpSCplx, const stpSCplx)

Complex exp function (e raised to y).

- void [CplxExp](#) (stpDCplx, const stpDCplx)
- void [CplxLog](#) (stpSCplx, const stpSCplx)

Complex natural logarithm.

- void [CplxLog](#) (stpDCplx, const stpDCplx)
- void [CplxLog10](#) (stpSCplx, const stpSCplx)

Complex 10-base logarithm.

- void [CplxLog10](#) (stpDCplx, const stpDCplx)
- void [CplxPow](#) (stpSCplx, const stpSCplx, const stpSCplx)

Complex x raised to y.

- void [CplxPow](#) (stpDCplx, const stpDCplx, const stpDCplx)
- void [CplxRoot](#) (stpSCplx, const stpSCplx, const stpSCplx)
Complex root y of x.
- void [CplxRoot](#) (stpDCplx, const stpDCplx, const stpDCplx)
- void [CplxConj](#) (stpSCplx spCplx)
- void [CplxConj](#) (stpDCplx spCplx)
- void [CplxConj](#) (stpSCplx, const stpSCplx)
- void [CplxConj](#) (stpDCplx, const stpDCplx)
- double [Multiple](#) (long)
Return multiple of n.
- float [ModZeroBessel](#) (float)
Zero-order modified Bessel function of the first kind.
- double [ModZeroBessel](#) (double)
- float [ChebyshevPolynom](#) (float, float)
nth-order Chebyshev polynomial
- double [ChebyshevPolynom](#) (double, double)

Private Attributes

- long [lPrevSrcCount](#)
- long [lPrevDestCount](#)
- float [fPI](#)
- double [dPI](#)
- int [iFIRDlyIdx](#)
- long [lFIRLength](#)
- [clDSPAlloc FIRCoeff](#)
- [clDSPAlloc FIRBuf](#)
- [clDSPAlloc FIRWork](#)
- float [fpIIR_C](#) [5]
- float [fpIIR_X](#) [3]
- float [fpIIR_Y](#) [2]
- double [dpIIR_C](#) [5]
- double [dpIIR_X](#) [3]
- double [dpIIR_Y](#) [2]
- bool [bFFTInitialized](#)
- bool [bRealTransform](#)
- long [lFFTLLength](#)
- float [fFFTSscale](#)
- double [dFFTSscale](#)
- long * [lpSBitRevWork](#)
- long * [lpDBitRevWork](#)
- float * [fpCosSinTable](#)

- double * [dpCosSinTable](#)
- void * [vpSTfrm](#)
- void * [vpDTfrm](#)
- [cIDSPAlloc SBitRevWork](#)
- [cIDSPAlloc DBitRevWork](#)
- [cIDSPAlloc SCosSinTable](#)
- [cIDSPAlloc DCosSinTable](#)
- [cIDSPAlloc FFTBuf](#)
- [clTransformS Tfrm](#)

5.10.1 Detailed Description

Class of DSP operations.

5.10.2 Constructor & Destructor Documentation

5.10.2.1 [cIDSPOp::cIDSPOp\(\)](#)

5.10.2.2 [cIDSPOp::~~cIDSPOp\(\)](#)

5.10.3 Member Function Documentation

5.10.3.1 **INLINE void cIDSPOp::Cart2Polar (float *, float *, float, float)** [static, protected]

Cartesian to polar conversion.

$$V = \sqrt{\Re^2 + \Im^2}$$

$$\varphi = \arctan\left(\frac{\Im}{\Re}\right)$$

Parameters:

Magn Magnitude

Phase Phase

Real Real

Imag Imaginary

5.10.3.2 **INLINE void cIDSPOp::Cart2Polar (double *, double *, double, double)** [static, protected]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.3 **INLINE void cIDSPOp::Cart2Polar (float *, float *, const *stpSCplx*)**
[static, protected]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Parameters:

Magn Magnitude

Phase Phase

Cplx Cartesian

5.10.3.4 **INLINE void cIDSPOp::Cart2Polar (double *, double *, const *stpDCplx*)**
[static, protected]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.5 **INLINE void cIDSPOp::Cart2Polar (*stpSPolar*, const *stpSCplx*)**
[static, protected]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Parameters:

Polar Polar

Cplx Cartesian

5.10.3.6 **INLINE void cIDSPOp::Cart2Polar (*stpDPolar*, const *stpDCplx*)**
[static, protected]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.7 **INLINE void cIDSPOp::Cart2Polar (*utpSCoord*)** [static, protected]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Parameters:

Coord Polar & Cartesian

5.10.3.8 **INLINE void cIDSPOp::Cart2Polar** ([utpDCoord](#)) [static, protected]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.9 **INLINE void cIDSPOp::Polar2Cart** (float *, float *, float, float) [static, protected]

Polar to cartesian conversion.

$$\Re = V \cos(\varphi)$$

$$\Im = V \sin(\varphi)$$

Note:

See Cart2Polar for details

5.10.3.10 **INLINE void cIDSPOp::Polar2Cart** (double *, double *, double, double) [static, protected]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.11 **INLINE void cIDSPOp::Polar2Cart** ([stpSCplx](#), float, float) [static, protected]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.12 **INLINE void cIDSPOp::Polar2Cart** ([stpDCplx](#), double, double) [static, protected]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.13 **INLINE void cIDSPOp::Polar2Cart** ([stpSCplx](#), const *stpSPolar*) [static, protected]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.14 **INLINE void cIDSPOp::Polar2Cart (stpDCplx, const stpDPolar)**
[static, protected]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.15 **INLINE void cIDSPOp::Polar2Cart (utpSCoord)** [static, protected]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.16 **INLINE void cIDSPOp::Polar2Cart (utpDCoord)** [static, protected]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.17 **INLINE void cIDSPOp::CplxAdd (stpSCplx, const stpSCplx)**
[static, protected]

Complex addition.

$$(z_r, z_i) = (x_r + y_r, x_i + y_i)$$

Parameters:

CplxDest Source & destination

CplxSrc Source

5.10.3.18 **INLINE void cIDSPOp::CplxAdd (stpDCplx, const stpDCplx)**
[static, protected]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.19 **INLINE void cIDSPOp::CplxAdd (stpSCplx, const stpSCplx, const stpSCplx)** [static, protected]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Parameters:

CplxDest Destination

CplxSrc1 Source 1

CplxSrc2 Source 2

5.10.3.20 **INLINE void cIDSPOp::CplxAdd** ([stpDCplx](#), const *stpDCplx*, const *stpDCplx*) [static, protected]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.21 **INLINE void cIDSPOp::CplxSub** ([stpSCplx](#), const *stpSCplx*) [static, protected]

Complex subtraction.

$$(z_r, z_i) = (x_r - y_r, x_i - y_i)$$

Parameters:

CplxDest Source & destination

CplxSrc Source

5.10.3.22 **INLINE void cIDSPOp::CplxSub** ([stpDCplx](#), const *stpDCplx*) [static, protected]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.23 **INLINE void cIDSPOp::CplxSub** ([stpSCplx](#), const *stpSCplx*, const *stpSCplx*) [static, protected]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Parameters:

CplxDest Destination

CplxSrc1 Source 1

CplxSrc2 Source 2

5.10.3.24 **INLINE void cIDSPOp::CplxSub** ([stpDCplx](#), const *stpDCplx*, const *stpDCplx*) [static, protected]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.25 **INLINE void cIDSPOp::CplxMul (stpSCplx, float)** [static, protected]

Complex multiply with real value.

$$(z_r, z_i) = (x_r y, x_i y)$$

Parameters:

CplxDest Source & destination

Src Source

5.10.3.26 **INLINE void cIDSPOp::CplxMul (stpDCplx, double)** [static, protected]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.27 **INLINE void cIDSPOp::CplxMul (stpSCplx, const stpSCplx)**
[static, protected]

Complex multiply.

$$\Re_z = \Re_x \Re_y - \Im_x \Im_y$$

$$\Im_z = \Re_x \Im_y + \Re_y \Im_x$$

Parameters:

CplxDest Source & destination

CplxSrc Source

5.10.3.28 **INLINE void cIDSPOp::CplxMul (stpDCplx, const stpDCplx)**
[static, protected]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.29 **INLINE void cIDSPOp::CplxMul (stpSCplx, const stpSCplx, const stpSCplx)** [static, protected]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Parameters:

CplxDest Destination

CplxSrc Source 1

CplxSrc Source 2

5.10.3.30 **INLINE** void **clDSPOp::CplxMul** (**stpDCplx**, const **stpDCplx**, const **stpDCplx**) [static, protected]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.31 **INLINE** void **clDSPOp::CplxMulC** (**stpSCplx**, const **stpSCplx**) [static, protected]

Complex multiply with complex conjugate.

$$\Re_z = \Re_x \Re_y - \Im_x (-\Im_y)$$

$$\Im_z = \Re_x (-\Im_y) + \Re_y \Im_x$$

Parameters:

CplxDest Source & destination

CplxSrc Source

5.10.3.32 **INLINE** void **clDSPOp::CplxMulC** (**stpDCplx**, const **stpDCplx**) [static, protected]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.33 **INLINE** void **clDSPOp::CplxMulC** (**stpSCplx**, const **stpSCplx**, const **stpSCplx**) [static, protected]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Parameters:

CplxDest Destination

CplxSrc1 Source 1

CplxSrc2 Source 2

5.10.3.34 **INLINE** void **clDSPOp::CplxMulC** (**stpDCplx**, const **stpDCplx**, const **stpDCplx**) [static, protected]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.335 **INLINE void cIDSPOp::CplxDiv** (**stpSCplx**, const *stpSCplx*) [static, protected]

Complex division.

$$\Re_z = \frac{\Re_x \Re_y + \Im_x \Im_y}{\Re_y^2 + \Im_y^2}$$

$$\Im_z = \frac{\Im_x \Re_y - \Re_x \Im_y}{\Re_y^2 + \Im_y^2}$$

Parameters:

CplxDest Source & destination

CplxSrc Source

5.10.336 **INLINE void cIDSPOp::CplxDiv** (**stpDCplx**, const *stpDCplx*) [static, protected]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.337 **INLINE void cIDSPOp::CplxDiv** (**stpSCplx**, const *stpSCplx*, const *stpSCplx*) [static, protected]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Parameters:

CplxDest Destination

CplxSrc Source 1

CplxSrc Source 2

5.10.338 **INLINE void cIDSPOp::CplxDiv** (**stpDCplx**, const *stpDCplx*, const *stpDCplx*) [static, protected]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.339 **INLINE void cIDSPOp::CplxExp** (**stpSCplx**, const *stpSCplx*) [static, protected]

Complex exp function (e raised to y).

$$\Re_z = \exp(\Re_y) \cos(\Im_y)$$

$$\Im_z = \exp(\Re_y) \sin(\Im_y)$$

Parameters:*CplxDest* Destination*CplxSrc* Source

5.10.3.40 **INLINE void** **clDSPOp::CplxExp** (**stpDCplx**, **const stpDCplx**)
[static, protected]

5.10.3.41 **INLINE void** **clDSPOp::CplxLog** (**stpSCplx**, **const stpSCplx**)
[static, protected]

Complex natural logarithm.

$$\Re_z = \ln(|y|)$$

$$\Im_z = \arg(y)$$

5.10.3.42 **INLINE void** **clDSPOp::CplxLog** (**stpDCplx**, **const stpDCplx**)
[static, protected]

5.10.3.43 **INLINE void** **clDSPOp::CplxLog10** (**stpSCplx**, **const stpSCplx**)
[static, protected]

Complex 10-base logarithm.

$$\Re_z = \log(|y|)$$

$$\Im_z = \arg(y)$$

5.10.3.44 **INLINE void** **clDSPOp::CplxLog10** (**stpDCplx**, **const stpDCplx**)
[static, protected]

5.10.3.45 **INLINE void** **clDSPOp::CplxPow** (**stpSCplx**, **const stpSCplx**, **const stpSCplx**) [static, protected]

Complex x raised to y.

$$z = \exp(\ln(x)y)$$

5.10.3.46 **INLINE** void cIDSPOp::CplxPow ([stpDCplx](#), const *stpDCplx*, const *stpDCplx*) [static, protected]

5.10.3.47 **INLINE** void cIDSPOp::CplxRoot ([stpSCplx](#), const *stpSCplx*, const *stpSCplx*) [static, protected]

Complex root y of x.

$$z = x^{\frac{1}{y}}$$

5.10.3.48 **INLINE** void cIDSPOp::CplxRoot ([stpDCplx](#), const *stpDCplx*, const *stpDCplx*) [static, protected]

5.10.3.49 void cIDSPOp::CplxConj ([stpSCplx](#) *spCplx*) [inline, static, protected]

5.10.3.50 void cIDSPOp::CplxConj ([stpDCplx](#) *spCplx*) [inline, static, protected]

5.10.3.51 **INLINE** void cIDSPOp::CplxConj ([stpSCplx](#), const *stpSCplx*) [static, protected]

5.10.3.52 **INLINE** void cIDSPOp::CplxConj ([stpDCplx](#), const *stpDCplx*) [static, protected]

5.10.3.53 **INLINE** double cIDSPOp::Multiple (long) [static, protected]

Return multiple of n.

$$!n$$

Parameters:

Value Value of n

5.10.3.54 **INLINE** float cIDSPOp::ModZeroBessel (float) [static, protected]

Zero-order modified Bessel function of the first kind.

$$I_0(x) = \sum_{k=0}^K \left[\frac{(x/2)^k}{k!} \right]^2$$

Parameters:

Value Value of x

5.10.3.55 **INLINE double cDSPOp::ModZeroBessel (double)** [static, protected]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.56 **float cDSPOp::ChebyshevPolynom (float, float)** [static, protected]

nth-order Chebyshev polynomial

$$T_n(x) = \begin{cases} \cos(n \cos^{-1} x), & |x| \leq 1 \\ \cosh(n \cosh^{-1} x), & |x| > 1 \end{cases}$$

Parameters:

Order Order of polynomial

Value Value of x

5.10.3.57 **double cDSPOp::ChebyshevPolynom (double, double)** [static, protected]

5.10.3.58 **signed long cDSPOp::Round (float)** [static]

Round floatingpoint number to integer.

5.10.3.59 **signed long cDSPOp::Round (double)** [static]

5.10.3.60 **void cDSPOp::Add (float *, float, long)** [static]

Add single value to vector.

$$x(i) = x(i) + y, 0 \leq i \leq N - 1$$

Parameters:

Dest Source & destination, x()

Src Source, y

Count Vector length, N

5.10.3.61 **void cDSPOp::Add (double *, double, long)** [static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.62 void cIDSPOp::Add (stpSCplx, stSCplx, long) [static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.63 void cIDSPOp::Add (stpDCplx, stDCplx, long) [static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.64 void cIDSPOp::Add (float *, const float *, long) [static]

Add two vectors.

$$x(i) = x(i) + y(i), 0 \leq i \leq N - 1$$

Parameters:

Dest Source & destination, x()

Src Source, y()

Count Vector length, N

5.10.3.65 void cIDSPOp::Add (double *, const double *, long) [static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.66 void cIDSPOp::Add (stpSCplx, const stpSCplx, long) [static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.67 void cIDSPOp::Add (stpDCplx, const stpDCplx, long) [static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.68 void cIDSPOp::Add (float *, const float *, const float *, long) [static]

Add two vectors and return result in third.

$$z(i) = x(i) + y(i), 0 \leq i \leq N - 1$$

Parameters:

Dest Destination, z()
Src1 Source 1, x()
Src2 Source 2, y()
Count Vector length, N

5.10.3.69 void cDSPop::Add (double *, const double *, const double *, long)
 [static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.70 void cDSPop::Add (stpSCplx, const stpSCplx, const stpSCplx, long)
 [static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.71 void cDSPop::Add (stpDCplx, const stpDCplx, const stpDCplx, long)
 [static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.72 void cDSPop::Sub (float *, float, long) [static]

Subtract single value from vector.

$$x(i) = x(i) - y, 0 \leq i \leq N - 1$$

Parameters:

Dest Source & destination, x()
Src Source, y
Count Vector length, N

5.10.3.73 void cDSPop::Sub (double *, double, long) [static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.74 void cDSPop::Sub (stpSCplx, stSCplx, long) [static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.75 void cIDSPOp::Sub (stpDCplx, stDCplx, long) [static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.76 void cIDSPOp::Sub (float *, const float *, long) [static]

Subtract two vectors.

$$x(i) = x(i) - y(i), 0 \leq i \leq N - 1$$

Parameters:

Dest Source & destination, x()

Src Source, y()

Count Vector length, N

5.10.3.77 void cIDSPOp::Sub (double *, const double *, long) [static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.78 void cIDSPOp::Sub (stpSCplx, const stpSCplx, long) [static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.79 void cIDSPOp::Sub (stpDCplx, const stpDCplx, long) [static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.80 void cIDSPOp::Sub (float *, const float *, const float *, long) [static]

Subtract two vectors and return result in third.

$$z(i) = x(i) - y(i), 0 \leq i \leq N - 1$$

Parameters:

Dest Destination, z()

Src1 Source 1, x()

Src2 Source 2, y()

Count Vector length, N

5.10.3.81 void cDSPOp::Sub (double *, const double *, const double *, long)
[static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.82 void cDSPOp::Sub (stpSCplx, const stpSCplx, const stpSCplx, long)
[static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.83 void cDSPOp::Sub (stpDCplx, const stpDCplx, const stpDCplx, long)
[static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.84 void cDSPOp::Mul (float *, float, long) [static]

Multiply vector with single value in-place.

$$x(i) = x(i)y, 0 \leq i \leq N - 1$$

Parameters:

Dest Source & destination, x()

Src Source, y

Count Vector length, N

5.10.3.85 void cDSPOp::Mul (double *, double, long) [static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.86 void cDSPOp::Mul (stpSCplx, float, long) [static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.87 void cDSPOp::Mul (stpDCplx, double, long) [static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.88 void cIDSPOp::Mul (stpSCplx, stSCplx, long) [static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.89 void cIDSPOp::Mul (stpDCplx, stDCplx, long) [static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.90 void cIDSPOp::Mul (float *, const float *, float, long) [static]

Multiply vector with single value out-of-place.

$$z(i) = x(i)y, 0 \leq i \leq N - 1$$

Parameters:

Dest Destination, z()

Src1 Source 1, x()

Src2 Source 2, y

Count Vector length, N

5.10.3.91 void cIDSPOp::Mul (double *, const double *, double, long) [static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.92 void cIDSPOp::Mul (float *, const float *, long) [static]

Multiply two vectors.

$$x(i) = x(i)y(i), 0 \leq i \leq N - 1$$

Parameters:

Dest Source & destination, x()

Src Source, y()

Count Vector length, N

5.10.3.93 void cIDSPOp::Mul (double *, const double *, long) [static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.94 void cIDSPOp::Mul (stpSCplx, const float *, long) [static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.95 void cIDSPOp::Mul (stpDCplx, const double *, long) [static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.96 void cIDSPOp::Mul (stpSCplx, const stpSCplx, long) [static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.97 void cIDSPOp::Mul (stpDCplx, const stpDCplx, long) [static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.98 void cIDSPOp::Mul (float *, const float *, const float *, long) [static]

Multiply two vectors and return result in third.

$$z(i) = x(i)y(i), 0 \leq i \leq N - 1$$

Parameters:

Dest Destination, z()

Src1 Source 1, x()

Src2 Source 2, y()

Count Vector length, N

5.10.3.99 void cIDSPOp::Mul (double *, const double *, const double *, long) [static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.100 void cIDSPOp::Mul (stpSCplx, const stpSCplx, const stpSCplx, long) [static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.101 void cIDSPOp::Mul (**stpDCplx**, const *stpDCplx*, const *stpDCplx*, long) [static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.102 void cIDSPOp::MulC (**stpSCplx**, const *stpSCplx*, long) [static]

Multiply vector with others complex conjugate.

$$X(i) = X(i)Y(i)^*, 0 \leq i \leq N - 1$$

$$Z(i) = X(i)Y(i)^*, 0 \leq i \leq N - 1$$

Parameters:

Dest Source & destination, X()

Src Source, Y()

Count Vector length, N

5.10.3.103 void cIDSPOp::MulC (**stpDCplx**, const *stpDCplx*, long) [static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.104 void cIDSPOp::MulC (**stpSCplx**, const *stpSCplx*, const *stpSCplx*, long) [static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.105 void cIDSPOp::MulC (**stpDCplx**, const *stpDCplx*, const *stpDCplx*, long) [static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.106 void cIDSPOp::Mul2 (float *, float *, const float *, long) [static]

Multiply two vectors in-place with third vector.

$$\begin{cases} x(i) = x(i)z(i) \\ y(i) = y(i)z(i) \end{cases}, 0 \leq i \leq N - 1$$

Parameters:

Dest1 Source & destination 1, x()

Dest2 Source & destination 2, y()

Src Source, z()

Count Vector length, N

5.10.3.107 void cIDSPOp::Mul2 (double *, double *, const double *, long)
[static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.108 void cIDSPOp::Mul2 (float *, float *, const float *, const float *, const float *, long) [static]

Multiply two vectors out-of-place with third vector and return results in separate two vectors.

Parameters:

Dest1 Destination 1

Dest2 Destination 2

Src1 Source 1

Src2 Source 2

Count Vector length

5.10.3.109 void cIDSPOp::Mul2 (double *, double *, const double *, const double *, const double *, long) [static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.110 void cIDSPOp::Div (float *, float, long) [static]

Divide vector with single value.

$$x(i) = \frac{x(i)}{y}, 0 \leq i \leq N - 1$$

Parameters:

Dest Source & destination, x()

Src Source, y

Count Vector length, N

5.10.3.111 void cIDSPOp::Div (double *, double, long) [static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.112 void cIDSPOp::Div (stpSCplx, stSCplx, long) [static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.113 void cIDSPOp::Div (stpDCplx, stDCplx, long) [static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.114 void cIDSPOp::Div (float *, const float *, long) [static]

Divide two vectors.

$$x(i) = \frac{x(i)}{y(i)}, 0 \leq i \leq N - 1$$

Parameters:

Dest Source & destination, x()

Src Source, y()

Count Vector length, N

5.10.3.115 void cIDSPOp::Div (double *, const double *, long) [static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.116 void cIDSPOp::Div (stpSCplx, const stpSCplx, long) [static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.117 void cIDSPOp::Div (stpDCplx, const stpDCplx, long) [static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.118 void cIDSPOp::Div (float *, const float *, const float *, long)
[static]

Divide two vectors and return result in third.

$$z(i) = \frac{x(i)}{y(i)}, 0 \leq i \leq N - 1$$

Parameters:

Dest Destination, z()

Src1 Source 1, x()

Src2 Source 2, y()

Count Vector length, N

5.10.3.119 void cIDSPOp::Div (double *, const double *, const double *, long)
[static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.120 void cIDSPOp::Div (stpSCplx, const stpSCplx, const stpSCplx, long)
[static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.121 void cIDSPOp::Div (stpDCplx, const stpDCplx, const stpDCplx, long)
[static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.122 void cIDSPOp::Div1x (float *, long) [static]

Calculate 1 / value in-place.

$$x(i) = \frac{1}{x(i)}, 0 \leq i \leq N - 1$$

Parameters:

Vect Source & destination

Count Vector length, N

5.10.3.123 void cDSPOp::Div1x (double *, long) [static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.124 void cDSPOp::Div1x (float *, const float *, long) [static]

Calculate 1 / value out-of-place.

$$y(i) = \frac{1}{x(i)}, 0 \leq i \leq N - 1$$

Parameters:

Dest Destination, y()

Src Source, x()

Count Vector length, N

5.10.3.125 void cDSPOp::Div1x (double *, const double *, long) [static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.126 void cDSPOp::MulAdd (float *, float, float, long) [static]

Multiply-add vector in-place.

$$x(i) = x(i)s + o, 0 \leq i \leq N - 1$$

Parameters:

Vect Source & destination, x()

Mul Multiply value, s

Add Addition value, o

Count Vector length, N

5.10.3.127 void cDSPOp::MulAdd (double *, double, double, long) [static]**5.10.3.128 void cDSPOp::MulAdd (float *, const float *, float, float, long) [static]**

Multiply-add vector out-of-place.

$$y(i) = x(i)s + o, 0 \leq i \leq N - 1$$

Parameters:

Dest Destination, y()
Src Source, x()
Mul Multiply value, s
Add Addition value, o
Count Vector length, N

5.10.3.129 `void cIDSPOp::MulAdd (double *, const double *, double, double, long) [static]`

5.10.3.130 `void cIDSPOp::Abs (float *, long) [static]`

Get absolute value in-place.

$$x(i) = |x(i)|, 0 \leq i \leq N - 1$$

Parameters:

Vect Source & destination, x()
Count Vector length, N

5.10.3.131 `void cIDSPOp::Abs (double *, long) [static]`

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.132 `void cIDSPOp::Abs (float *, const float *, long) [static]`

Get absolute value out-of-place.

$$y(i) = |x(i)|, 0 \leq i \leq N - 1$$

Parameters:

Dest Destination, y()
Src Source, x()
Count Vector length, N

5.10.3.133 `void cIDSPOp::Abs (double *, const double *, long) [static]`

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.134 void cIDSPOp::Sqrt (float *, long) [static]

Calculate square-roots in-place.

$$x(i) = \sqrt{x(i)}, 0 \leq i \leq N - 1$$

Parameters:

Vect Source & destination, x()

Count Vector length, N

5.10.3.135 void cIDSPOp::Sqrt (double *, long) [static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.136 void cIDSPOp::Sqrt (float *, const float *, long) [static]

Calculate square-roots out-of-place.

$$y(i) = \sqrt{x(i)}, 0 \leq i \leq N - 1$$

Parameters:

Dest Destination, y()

Src Source, x()

Count Vector length, N

5.10.3.137 void cIDSPOp::Sqrt (double *, const double *, long) [static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.138 void cIDSPOp::Zero (float *, long) [static]

Set all vector elements to zero.

Parameters:

Vect Vector

Count Vector length

5.10.3.139 void cIDSPOp::Zero (double *, long) [static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.140 void cDSPOp::Zero (stpSCplx, long) [static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.141 void cDSPOp::Zero (stpDCplx, long) [static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.142 void cDSPOp::Set (float *, float, long) [static]

Set all vector elements to specified value.

Parameters:

Dest Destination
Src Source value
Count Vector length

5.10.3.143 void cDSPOp::Set (double *, double, long) [static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.144 void cDSPOp::Set (stpSCplx, stSCplx, long) [static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.145 void cDSPOp::Set (stpDCplx, stDCplx, long) [static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.146 void cDSPOp::Set (float *, float, long, long, long) [static]

Set specified vector elements to specified value with boundary check.

Parameters:

Dest Destination
Src Source value
Start Starting index
Count Number of elements to set
Length Vector length

5.10.3.147 void cIDSPOp::Set (double *, double, long, long, long) [static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.148 void cIDSPOp::Set (stpSCplx, stSCplx, long, long, long) [static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.149 void cIDSPOp::Set (stpDCplx, stDCplx, long, long, long) [static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.150 void cIDSPOp::Clip (float *, float, long) [static]

Clip vector in-place to specified value.

$$x(i) = \begin{cases} x(i) & , x(i) \leq y \\ y & , x(i) > y \end{cases} , 0 \leq i \leq N - 1$$

Parameters:

Vect Source & destination, x()
Value Clipping value, y
Count Vector length, N

5.10.3.151 void cIDSPOp::Clip (double *, double, long) [static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.152 void cIDSPOp::Clip (float *, const float *, float, long) [static]

Clip vector out-of-place to specified value.

$$z(i) = \begin{cases} x(i) & , x(i) \leq y \\ y & , x(i) > y \end{cases} , 0 \leq i \leq N - 1$$

Parameters:

Dest Destination, z()
Src Source, x()
Value Clipping value, y
Count Vector length, N

5.10.3.153 void cDSPOp::Clip (double *, const double *, double, long) [static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.154 void cDSPOp::Clip (float *, float, float, long) [static]

Clip vector in-place to fit between specified values.

$$x(i) = \begin{cases} a & , x(i) < a \\ x(i) & , a \leq x(i) \leq b \\ b & , x(i) > b \end{cases} \quad , 0 \leq i \leq N - 1$$

Parameters:

Vect Source & destination, x()

Min Minimum value, a

Max Maximum value, b

Count Vector length, N

5.10.3.155 void cDSPOp::Clip (double *, double, double, long) [static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.156 void cDSPOp::Clip (float *, const float *, float, float, long) [static]

Clip vector out-of-place to fit between specified values.

$$y(i) = \begin{cases} a & , x(i) < a \\ x(i) & , a \leq x(i) \leq b \\ b & , x(i) > b \end{cases} \quad , 0 \leq i \leq N - 1$$

Parameters:

Dest Destination, y()

Src Source, x()

Min Minimum value, a

Max Maximum value, b

Count Vector length, N

5.10.3.157 void cIDSPOp::Clip (double *, const double *, double, double, long) [static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.158 void cIDSPOp::ClipZero (float *, long) [static]

Clip smallest values in vector to zero (in-place).

$$x(i) = \begin{cases} 0 & , x(i) < 0 \\ x(i) & , x(i) \leq 0 \end{cases} , 0 \leq i \leq N - 1$$

Parameters:

Vect Source & destination, x()

Count Vector length, N

5.10.3.159 void cIDSPOp::ClipZero (double *, long) [static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.160 void cIDSPOp::ClipZero (float *, const float *, long) [static]

Clip smallest values in vector to zero (out-of-place).

$$y(i) = \begin{cases} 0 & , x(i) < 0 \\ x(i) & , x(i) \leq 0 \end{cases} , 0 \leq i \leq N - 1$$

Parameters:

Dest Destination, y()

Src Source, x()

Count Vector length, N

5.10.3.161 void cIDSPOp::ClipZero (double *, const double *, long) [static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.162 void cDSPOp::Copy (float *, const float *, long) [static]

Copy source vector to destination vector(s).

Parameters:

Dest Destination vector

Src Source vector

Count Vector length

5.10.3.163 void cDSPOp::Copy (double *, const double *, long) [static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.164 void cDSPOp::Copy (stpSCplx, const stpSCplx, long) [static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.165 void cDSPOp::Copy (stpDCplx, const stpDCplx, long) [static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.166 void cDSPOp::Copy (float *, float *, const float *, long) [static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.167 void cDSPOp::Copy (double *, double *, const double *, long) [static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.168 float cDSPOp::Convolve (const float *, const float *, long) [static]

Convolve two vectors (same length, finite) without lag.

Note:

Circular convolution, result is not scaled

$$z = \sum_{i=0}^K x(i)y(K-i), K = N-1$$

Parameters:**Src1** Source 1, x()**Src2** Source 2, y()**Count** Vector length, N**Returns:**

Convolution result, z

5.10.3.169 double cIDSPOp::Convolve (const double *, const double *, long)
`[static]`

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.170 void cIDSPOp::Convolve (float *, const float *, const float *, long)
`[static]`

Convolve two vectors (same length, finite) with lag.

Note:

Circular convolution, result is not scaled

$$z(i) = \sum_{k=0}^{N-1} x(i-k)y(k)$$

Parameters:**Dest** Destination, z()**Src1** Source 1, x()**Src2** Source 2, y()**Count** Vector length, N

5.10.3.171 void cIDSPOp::Convolve (double *, const double *, const double *, long)
`[static]`

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.172 **float cIDSPOp::Correlate (const float *, const float *, long)** [static]

Correlate two vectors (same length, finite) without lag.

Note:

Circular correlation, result is scaled

$$z = \frac{1}{N} \sum_{i=0}^{N-1} x(i)y(i)$$

Parameters:

Src1 Source1, x()

Src2 Source2, y()

Count Vector length, N

Returns:

Correlation result, z

5.10.3.173 **double cIDSPOp::Correlate (const double *, const double *, long)** [static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.174 **void cIDSPOp::Correlate (float *, const float *, const float *, long)** [static]

Correlate two vectors (same length, finite) with lag.

Note:

Circular correlation, result is scaled

$$z(i) = \frac{1}{N} \sum_{k=0}^{N-1} x(k)y(k+i)$$

Parameters:

Dest Destination, z()

Src1 Source 1, x()

Src2 Source 2, y()

Count Vector length, N

5.10.3.175 void cIDSPOp::Correlate (double *, const double *, const double *, long) [static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.176 float cIDSPOp::AutoCorrelate (const float *, long) [static]

Autocorrelate vector.

Note:

Circular, result is scaled

$$z = \frac{1}{N} \sum_{i=0}^{N-1} x(i)^2$$

Parameters:

Src Source, x()

Count Vector length, N

Returns:

Autocorrelation (energy)

5.10.3.177 double cIDSPOp::AutoCorrelate (const double *, long) [static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.178 void cIDSPOp::AutoCorrelate (float *, const float *, long) [static]

Autocorrelate vector.

Note:

Circular, result is scaled

$$y(i) = \frac{1}{N} \sum_{k=0}^{N-1} x(k)x(k+i)$$

Parameters:

Dest Destination, y()

Src Source, x()

Count Vector length, N

5.10.3.179 void cIDSPOp::AutoCorrelate (double *, const double *, long)
[static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.180 float cIDSPOp::DotProduct (const float *, const float *, long)
[static]

Dot product of two vectors.

$$z = \sum_{i=0}^{N-1} x(i)y(i)$$

Parameters:

Src1 Source 1, x()
Src2 Source 2, y()
Count Vector length, N

Returns:

Dot product

5.10.3.181 double cIDSPOp::DotProduct (const double *, const double *, long)
[static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.182 void cIDSPOp::MinMax (float *, float *, const float *, long)
[static]

Find minimum and maximum values of vector.

Parameters:

Min Minimum
Max Maximum
Src Source
Count Vector length

5.10.3.183 void cIDSPOp::MinMax (double *, double *, const double *, long)
[static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.184 float cIDSPOp::Mean (const float *, long) [static]

Calculate mean of vector.

$$a = \frac{1}{N} \sum_{i=0}^{N-1} x(i)$$

Parameters:

Src Source

Count Vector length

Returns:

Mean

5.10.3.185 double cIDSPOp::Mean (const double *, long) [static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.186 float cIDSPOp::Median (const float *, long) [static]

Calculate median of vector.

$$m = \begin{cases} x_{sorted}(\frac{N}{2}) & , N \bmod 2 \neq 0 \\ 0.5 \times (x_{sorted}(\frac{N}{2} - 1) + x_{sorted}(\frac{N}{2})) & , N \bmod 2 = 0 \end{cases}$$

Parameters:

Src Source, x()

Count Vector length, N

Returns:

Median, m

5.10.3.187 double cIDSPOp::Median (const double *, long) [static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.188 void cIDSPOp::Negate (float *, long) [static]

Negate vector elements.

$$x(i) = -x(i), 0 \leq i \leq N - 1$$

Parameters:*Vect* Source & destination*Count* Vector length**5.10.3.189 void cDSPOp::Negate (double *, long) [static]**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.190 void cDSPOp::Negate (float *, const float *, long) [static]

Negate vector elements.

$$y(i) = -x(i), 0 \leq i \leq N - 1$$

Parameters:*Dest* Destination, y()*Src* Source, x()*Count* Vector length, N**5.10.3.191 void cDSPOp::Negate (double *, const double *, long) [static]**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.192 void cDSPOp::Normalize (float *, long) [static]

Normalize vector elements.

$$x(i) = \frac{x(i) - \mu}{\sigma}, 0 \leq i \leq N - 1$$

Parameters:*Vect* Source & destination, x()*Count* Vector length, N**5.10.3.193 void cDSPOp::Normalize (double *, long) [static]**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.194 void cIDSPOp::Normalize (float *, const float *, long) [static]

Normalize vector elements.

$$y(i) = \frac{x(i) - \mu}{\sigma}, 0 \leq i \leq N - 1$$

Parameters:

Dest Destination, y()

Src Source, x()

Count Vector length, N

5.10.3.195 void cIDSPOp::Normalize (double *, const double *, long) [static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.196 float cIDSPOp::Product (const float *, long) [static]

Product of vector elements.

$$p = \prod x(i), 0 \leq i \leq N - 1$$

Parameters:

Src Source, x()

Count Vector length, N

Returns:

Product, p

5.10.3.197 double cIDSPOp::Product (const double *, long) [static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.198 void cIDSPOp::Reverse (float *, long) [static]

Reverse vector (in-place).

$$y(i) = x(K - i), K = N - 1, 0 \leq i \leq N - 1$$

Parameters:

Vect Source & destination x()

Count Vector length, N

5.10.3.199 void cDSPOp::Reverse (double *, long) [static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.200 void cDSPOp::Reverse (stpSCplx, long) [static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.201 void cDSPOp::Reverse (stpDCplx, long) [static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.202 void cDSPOp::Reverse (float *, const float *, long) [static]

Reverse vector (out-of-place).

$$y(i) = x(K - i), K = N - 1, 0 \leq i \leq N - 1$$

Parameters:

Dest Destination, y()

Src Source, x()

Count Vector length

5.10.3.203 void cDSPOp::Reverse (double *, const double *, long) [static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.204 void cDSPOp::Reverse (stpSCplx, const stpSCplx, long) [static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.205 void cDSPOp::Reverse (stpDCplx, const stpDCplx, long) [static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.206 void cIDSPOp::Scale (float *, long) [static]

Scale (normalize) vector to range [-1:1].

$$x(i) = x(i) \frac{2}{\max - \min} + \left(1 - \max \frac{2}{\max - \min}\right), 0 \leq i \leq N - 1$$

Parameters:

Vect Source & destination, x()

Count Vector length, N

5.10.3.207 void cIDSPOp::Scale (double *, long) [static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.208 void cIDSPOp::Scale (float *, const float *, long) [static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Parameters:

Dest Destination

Src Source

Count Vector length

5.10.3.209 void cIDSPOp::Scale (double *, const double *, long) [static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.210 void cIDSPOp::Scale01 (float *, long) [static]

Scale (normalize) vector to range [0:1].

$$x(i) = x(i) \frac{1}{\max - \min} + \left(-\min \frac{1}{\max - \min}\right), 0 \leq i \leq N - 1$$

Parameters:

Vect Source & destination, x()

Count Vector length, N

5.10.3.211 void cDSPOp::Scale01 (double *, long) [static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.212 void cDSPOp::Scale01 (float *, const float *, long) [static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Parameters:

Dest Destination

Src Source

Count Vector length

5.10.3.213 void cDSPOp::Scale01 (double *, const double *, long) [static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.214 void cDSPOp::Sort (float *, long) [static]

Sort vector elements (in-place).

Vector is sorted using quick-sort algorithm.

Parameters:

Vect Source & destination

Count Vector length

5.10.3.215 void cDSPOp::Sort (double *, long) [static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.216 void cDSPOp::Sort (long *, long) [static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.217 void cIDSPOp::StdDev (float *, float *, const float *, long) [static]

Calculate standard deviation and mean of vector.

$$\mu = \frac{1}{N} \sum_{i=0}^{N-1} x(i)$$

$$\sigma = \sqrt{\frac{1}{N} \sum_{i=0}^{N-1} (x(i) - \mu)^2}$$

Parameters:

StdDev Standard deviation, σ

Mean Mean, μ

Src Source, x()

Count Vector length, N

5.10.3.218 void cIDSPOp::StdDev (double *, double *, const double *, long) [static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.219 float cIDSPOp::Sum (const float *, long) [static]

Calculate sum of vector elements.

$$y = \sum_{i=0}^{N-1} x(i)$$

Parameters:

Src Source, x()

Count Vector length, N

Returns:

Sum, y

5.10.3.220 double cIDSPOp::Sum (const double *, long) [static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.221 void cDSPOp::Square (float *, long) [static]

Square vector in-place.

$$x(i) = x(i)^2, 0 \leq i \leq N - 1$$

Parameters:

Vect Source & destination, x()

Count Vector length, N

5.10.3.222 void cDSPOp::Square (double *, long) [static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.223 void cDSPOp::Square (float *, const float *, long) [static]

Square vector out-of-place.

$$y(i) = x(i)^2, 0 \leq i \leq N - 1$$

Parameters:

Dest Destination, y()

Src Source, x()

Count Vector length, N

5.10.3.224 void cDSPOp::Square (double *, const double *, long) [static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.225 void cDSPOp::Convert (float *, const unsigned char *, long) [static]

Vector datatype conversion.

Note:

12-bit data in LSBs, 24-bit data in MSBs

Parameters:

Dest Destination

Src Source

Count Vector length

5.10.3.226 void cIDSPOp::Convert (float *, const signed short *, long, bool)
[static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Parameters:

Dest Destination

Src Source

Count Vector length

12bit 12-bit data

5.10.3.227 void cIDSPOp::Convert (float *, const signed int *, long, bool)
[static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Parameters:

Dest Destination

Src Source

Count Vector length

24bit 24-bit data

5.10.3.228 void cIDSPOp::Convert (float *, const double *, long) [static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.229 void cIDSPOp::Convert (double *, const unsigned char *, long)
[static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.230 void cIDSPOp::Convert (double *, const signed short *, long, bool)
[static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.231 void cDSPOp::Convert (double *, const signed int *, long, bool)
[static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.232 void cDSPOp::Convert (double *, const float *, long) [static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.233 void cDSPOp::Convert (unsigned char *, const float *, long)
[static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.234 void cDSPOp::Convert (unsigned char *, const double *, long)
[static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.235 void cDSPOp::Convert (signed short *, const float *, long, bool)
[static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.236 void cDSPOp::Convert (signed short *, const double *, long, bool)
[static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.237 void cDSPOp::Convert (signed int *, const float *, long, bool)
[static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.238 void cIDSPOp::Convert (signed int *, const double *, long, bool)
[static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.239 void cIDSPOp::CartToPolar (float *, float *, const float *, const float *, long) [static]

Convert cartesian to polar vectors (out-of-place).

See ::Cart2Polar for formula

Parameters:

Magn Magnitude

Phase Phase

Real Real

Imag Imaginary

Count Vector length

5.10.3.240 void cIDSPOp::CartToPolar (double *, double *, const double *, const double *, long) [static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.241 void cIDSPOp::CartToPolar (float *, float *, const stpSCplx, long)
[static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Parameters:

Magn Magnitude

Phase Phase

Cart Cartesian

Count Vector length

5.10.3.242 void cIDSPOp::CartToPolar (double *, double *, const stpDCplx, long) [static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.243 void cIDSPOp::CartToPolar ([stpSPolar](#), const *stpSCplx*, long)
[static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Parameters:

Polar Polart
Cart Cartesian
Count Vector length

5.10.3.244 void cIDSPOp::CartToPolar ([stpDPolar](#), const *stpDCplx*, long)
[static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.245 void cIDSPOp::CartToPolar ([utpSCoord](#), long) [static]

Convert cartesian to polar vectors (in-place).

5.10.3.246 void cIDSPOp::CartToPolar ([utpDCoord](#), long) [static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.247 void cIDSPOp::PolarToCart (float *, float *, const float *, const float *, long) [static]

Convert polar to cartesian vectors (out-of-place).

See ::Polar2Cart for formula

Parameters:

Real Real
Imag Imaginary
Magn Magnitude
Phase Phase
Count Vector length

5.10.3.248 void cIDSPOp::PolarToCart (double *, double *, const double *, const double *, long) [static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.249 void cIDSPOp::PolarToCart ([stpSCplx](#), const float *, const float *, long) [static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Parameters:

Cart Cartesian

Magn Magnitude

Phase Phase

Count Vector length

5.10.3.250 void cIDSPOp::PolarToCart ([stpDCplx](#), const double *, const double *, long) [static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.251 void cIDSPOp::PolarToCart ([stpSCplx](#), const *stpSPolar*, long) [static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Parameters:

Polar Polar

Cart Cartesian

Count Vector length

5.10.3.252 void cIDSPOp::PolarToCart ([stpDCplx](#), const *stpDPolar*, long) [static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.253 void cIDSPOp::PolarToCart ([utpSCoord](#), long) [static]

Convert cartesian to polar vectors (in-place).

5.10.3.254 void cIDSPOp::PolarToCart ([utpDCoord](#), long) [static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.255 float cIDSPOp::CrossCorr (const float *, const float *, long)
[static]

Calculate normalized cross correlation of two vectors without delay.

$$z = \frac{\frac{1}{N} \sum_{i=0}^{N-1} x(i)y(i)}{\frac{1}{N} \sqrt{\sum_{i=0}^{N-1} x(i)^2 \sum_{i=0}^{N-1} y(i)^2}}$$

Parameters:

Src1 Source 1, x()

Src2 Source 2, y()

Count Vector length, N

Returns:

Normalized cross-correlation

5.10.3.256 double cIDSPOp::CrossCorr (const double *, const double *, long)
[static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.257 float cIDSPOp::DelCrossCorr (const float *, const float *, long, long)
[static]

Calculate normalized cross correlation of two vectors with delay in second.

$$z = \frac{\frac{1}{K} \sum_{i=0}^{K-1} x(i)y(i+k)}{\frac{1}{K} \sqrt{\sum_{i=0}^{K-1} x(i)^2 \sum_{i=0}^{K-1} y(i+k)^2}}, K = N - k$$

Parameters:

Src1 Source 1, x()

Src2 Source 2, y()

Delay Delay, k

Count Vector length, N

Returns:

Normalized cross-correlation

5.10.3.258 double cIDSPOp::DelCrossCorr (const double *, const double *, long, long)
[static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.259 `void cIDSPOp::DelCrossCorr (float *, const float *, const float *, long, const long *, long) [static]`

Calculate normalized cross correlation of two vectors with delays specified in vector.

$$z(j) = \frac{\frac{1}{K} \sum_{i=0}^{K-1} x(i)y(i+k(j))}{\frac{1}{K} \sqrt{\sum_{i=0}^{K-1} x(i)^2 \sum_{i=0}^{K-1} y(i+k(j))^2}}, K = N - k(j), 0 \leq j \leq M - 1$$

Parameters:

Dest Destination vector, z()

Src1 Source 1, x()

Src2 Source 2, y()

Count Vector length, N

Delay Delays, k()

DelayCount Delay vector length, M

5.10.3.260 `void cIDSPOp::DelCrossCorr (double *, const double *, const double *, long, const long *, long) [static]`

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.261 `float cIDSPOp::Energy (const float *, long) [static]`

Calculates energy (square) of vector.

$$w = \sum_{i=0}^{N-1} x(i)^2$$

Parameters:

Src Source, x()

Count Vector length, N

Returns:

Energy, w

5.10.3.262 `double cIDSPOp::Energy (const double *, long) [static]`

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.263 void cIDSPOp::Magnitude (float *, const *stpSCplx*, long)
[static]

Calculates magnitudes (linear) of vector.

$$V(i) = \sqrt{\Re_X(i)^2 + \Im_X(i)^2}, 0 \leq i \leq N - 1$$

Parameters:

Dest Destination, V()

Src Source, X()

Count Vector length

5.10.3.264 void cIDSPOp::Magnitude (double *, const *stpDCplx*, long)
[static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.265 void cIDSPOp::Power (float *, const *stpSCplx*, long) [static]

Calculates powers (in dB) of vector.

$$P(i) = 20 \log \sqrt{\Re_X(i)^2 + \Im_X(i)^2}, 0 \leq i \leq N - 1$$

Parameters:

Dest Destination, P()

Src Source, X()

Count Vector length

5.10.3.266 void cIDSPOp::Power (double *, const *stpDCplx*, long) [static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.267 void cIDSPOp::Phase (float *, const *stpSCplx*, long) [static]

Calculated phases (in rad) of vector.

$$\varphi(i) = \arctan \frac{\Im_X(i)}{\Re_X(i)}, 0 \leq i \leq N - 1$$

Parameters:

Dest Destination, $\varphi(i)$

Src Source, X()

Count Vector length, N

5.10.3.268 void cIDSPOp::Phase (double *, const *stpDCplx*, long) [static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.269 void cIDSPOp::PowerPhase (float *, float *, const *stpSCplx*, long) [static]

Calculates powers (in dB) and phases (in rad) of vector See ::Power() and ::Phase() for formulas.

Parameters:

Power Powers

Phase Phases

Src Source

Count Vector length

5.10.3.270 void cIDSPOp::PowerPhase (double *, double *, const *stpDCplx*, long) [static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.271 void cIDSPOp::Decimate (float *, const float *, long, long) [static]

Decimate vector without average.

Note:

This can be used in-place also

Parameters:

Dest Destination

Src Source

Factor Decimation factor

Count Vector length (source)

5.10.3.272 void cIDSPOp::Decimate (double *, const double *, long, long) [static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.273 void cDSPOp::DecimateAvg (float *, const float *, long, long)
[static]

Decimate vector with average.

Linear (arithmetic) mean is used to evaluate new values.

Parameters:

Dest Destination

Src Source

Factor Decimation factor

Count Vector length (source)

5.10.3.274 void cDSPOp::DecimateAvg (double *, const double *, long, long)
[static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.275 void cDSPOp::Interpolate (float *, const float *, long, long)
[static]

Interpolate vector without average (nulling).

Parameters:

Dest Destination

Src Source

Factor Interpolation factor

Count Vector length (source)

5.10.3.276 void cDSPOp::Interpolate (double *, const double *, long, long)
[static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.277 void cDSPOp::InterpolateAvg (float *, const float *, long, long)
[static]

Interpolate vector with average (linear).

Lagrange interpolation is used to estimate new values.

$$y(k) = \frac{k - (i + 1)}{i - (i + 1)} \times x(i) + \frac{k - i}{(i + 1) - i} \times x(i + 1), 0 \leq k \leq M - 1, 0 \leq i \leq N - 1$$

Where M is destination length and N is source length.

Parameters:

Dest Destination

Src Source

Factor Interpolation factor

Count Vector length (source)

5.10.3.278 `void cIDSPOp::InterpolateAvg (double *, const double *, long, long)`
[static]

5.10.3.279 `void cIDSPOp::Resample (float *, long, const float *, long)`
[static]

Resample vector to different length.

Parameters:

Dest Destination

DestCount Destination vector length

Src Source

SrcCount Source vector length

5.10.3.280 `void cIDSPOp::Resample (double *, long, const double *, long)`
[static]

5.10.3.281 `void cIDSPOp::ResampleAvg (float *, long, const float *, long)`
[static]

Resample vector to different length with average.

Parameters:

Dest Destination

DestCount Destination vector length

Src Source

SrcCount Source vector length

5.10.3.282 `void cIDSPOp::ResampleAvg (double *, long, const double *, long)`
[static]

5.10.3.283 `float cIDSPOp::RMS (const float *, long)` [static]

Calculate RMS (root mean square) of vector.

$$I = \sqrt{\frac{\sum_{i=0}^{N-1} x(i)^2}{N}}$$

Parameters:*Src* Source, x()*Count* Vector length, N**Returns:**

RMS, I

5.10.3.284 double cIDSPOp::RMS (const double *, long) [static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.285 float cIDSPOp::Variance (float *, float *, const float *, long) [static]

Calculate variance and mean of vector.

Note:

variance and mean pointers can be NULL

$$\mu = \frac{1}{N} \sum_{i=0}^{N-1} x(i)$$

$$\sigma^2 = \frac{\sum_{i=0}^N (x(i) - \mu)^2}{N}$$

Parameters:*Variance* Variance, σ^2 *Mean* Mean, μ *Src* Source, x()*Count* Vector length, N**Returns:**Variance σ^2 **5.10.3.286 double cIDSPOp::Variance (double *, double *, const double *, long) [static]**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.287 float cIDSPOp::PeakLevel (const float *, long) [static]

Find peak level of vector and return result in dB.

Parameters:

Src Source

Count Vector length

Returns:

Peak level (dB)

5.10.3.288 double cIDSPOp::PeakLevel (const double *, long) [static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.289 void cIDSPOp::WinBartlett (float *, long)

Calculate Bartlett (triangle) window coefficients.

Parameters:

Dest Destination

Count Vector length

5.10.3.290 void cIDSPOp::WinBartlett (double *, long)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.291 void cIDSPOp::WinBlackman (float *, long)

Calculate Blackman window coefficients.

$$y(n) = 0.42 - 0.5 \cos\left(\frac{2\pi n}{N}\right) + 0.08 \cos\left(\frac{4\pi n}{N}\right), 0 \leq n \leq N - 1$$

Parameters:

Dest Destination, y()

Count Vector length, N

5.10.3.292 void cIDSPOp::WinBlackman (double *, long)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.293 void cDSPOp::WinBlackman (float *, long, float)

Calculate blackman window coefficients with specified alpha.

$$y(n) = \frac{\alpha + 1}{2} - 0.5 \cos\left(\frac{2\pi n}{N-1}\right) - \frac{\alpha}{2} \cos\left(\frac{4\pi n}{N-1}\right), 0 \leq n \leq N-1$$

or with optimal alpha when alpha is specified as 0

$$\alpha = \frac{0.5}{1 + \cos\left(\frac{2\pi}{N-1}\right)}$$

(around -0.25 for large windows)

5.10.3.294 void cDSPOp::WinBlackman (double *, long, double)**5.10.3.295 void cDSPOp::WinBlackmanHarris (float *, long)**

Calculate Blackman-Harris window coefficients.

$$y(n) = 0.42323 - 0.49855 \cos\left(\frac{2\pi n}{N}\right) + 0.07922 \cos\left(\frac{4\pi n}{N}\right), 0 \leq n \leq N-1$$

Parameters:

Dest Destination, y()

Count Vector length, N

5.10.3.296 void cDSPOp::WinBlackmanHarris (double *, long)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.297 void cDSPOp::WinCosTapered (float *, long)

Calculate cosine tapered window coefficients.

$$y(n) = 0.5 \left(1 - \cos\left(\frac{2\pi n}{N}\right)\right), 0 \leq n \leq N-1$$

Parameters:

Dest Destination, y()

Count Vector length, N

5.10.3.298 void cIDSPOp::WinCosTapered (double *, long)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.299 void cIDSPOp::WinCosTaperedA (float *, long)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Apply cosine tapered window to vector (in-place)

5.10.3.300 void cIDSPOp::WinCosTaperedA (double *, long)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.301 void cIDSPOp::WinCosTaperedA (float *, const float *, long)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Apply cosine tapered window to vector (out-of-place)

5.10.3.302 void cIDSPOp::WinCosTaperedA (double *, const double *, long)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.303 void cIDSPOp::WinExactBlackman (float *, long)

Calculate exact Blackman window coefficients.

$$y(n) = \frac{7938}{18608} - \frac{9240}{18608} \cos\left(\frac{2\pi n}{N}\right) + \frac{1430}{18608} \cos\left(\frac{4\pi n}{N}\right), 0 \leq n \leq N - 1$$

Parameters:

Dest Destination, y()

Count Vector length, N

5.10.3.304 void cIDSPOp::WinExactBlackman (double *, long)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.305 void cDSPOp::WinExp (float *, float, long)

Calculate exponential window coefficients.

$$y(n) = y(N - 1 - n) = \exp\left(\frac{\ln(z + 1)}{N/2}n\right) - 1, 0 \leq n \leq N/2$$

Parameters:

Dest Destination, y()

Final Final value, z

Count Vector length, N

5.10.3.306 void cDSPOp::WinExp (double *, double, long)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.307 void cDSPOp::WinFlatTop (float *, long)

Calculate flat top window coefficients.

$$y(n) = 0.2810639 - 0.5208972 \cos\left(\frac{2\pi n}{N}\right) + 0.1980399 \cos\left(\frac{4\pi n}{N}\right), 0 \leq n \leq N - 1$$

Parameters:

Dest Destination, y()

Count Vector length, N

5.10.3.308 void cDSPOp::WinFlatTop (double *, long)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.309 void cDSPOp::WinGenericCos (float *, long, const float *, long)

Calculate generic cosine window coefficients.

$$y(n) = \sum_{i=0}^{M-1} -1^i x(i) \cos\left(\frac{2\pi i n}{N}\right), 0 \leq n \leq N - 1$$

Parameters:

Dest Destination, y()

Count Vector length, N

Coeff Coefficients, x()

CoeffCount Coefficient vector length, M

5.10.3.310 void cDSPop::WinGenericCos (double *, long, const double *, long)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.311 void cDSPop::WinHamming (float *, long)

Calculate Hamming window coefficients.

$$y(n) = 0.54 - 0.46 \cos\left(\frac{2\pi n}{N}\right), 0 \leq n \leq N - 1$$

Parameters:

Dest Destination vector, y()

Count Vector length, N

5.10.3.312 void cDSPop::WinHamming (double *, long)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.313 void cDSPop::WinHanning (float *, long)

Calculate Hanning (Hann?) window coefficients.

$$y(n) = 0.5 - 0.5 \cos\left(\frac{2\pi n}{N}\right), 0 \leq n \leq N - 1$$

Parameters:

Dest Destination vector, y()

Count Vector length, N

5.10.3.314 void cDSPop::WinHanning (double *, long)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.315 void cDSPop::WinKaiser (float *, float, long)

Calculate Kaiser window coefficients See ::ModZeroBessel() for I_0 .

$$y(n) = \frac{I_0 \left\{ \beta \sqrt{1 - \left| 1 - \frac{2n}{N} \right|^2} \right\}}{I_0 \{ \beta \}}, 0 \leq n \leq N - 1$$

Parameters:

Dest Destination, y()
Beta Beta, β
Count Vector length, N

5.10.3.316 void cDSPOp::WinKaiser (double *, double, long)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.317 void cDSPOp::WinKaiserBessel (float *, float, long)

Calculate Kaiser-Bessel window coefficients See ::ModZeroBessel() for I_0 .

$$w(n_{DFT}) = I_0 \left\{ \pi \alpha \left[1.0 - \left(\frac{n_{DFT} - N/2}{N/2} \right)^2 \right]^{1/2} \right\} / I_0(\pi \alpha), 0 \leq n_{DFT} \leq N - 1$$

Parameters:

Dest Destination, w()
Alpha Alpha, α
Count Vector length, N

5.10.3.318 void cDSPOp::WinKaiserBessel (double *, double, long)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.319 void cDSPOp::WinTukey (float *, long)

Calculate Tukey window coefficients.

$$y(n) = 0.5 \left[1 + \cos \left(\frac{(n - N/2)\pi}{N/2} \right) \right], 0 \leq n \leq N - 1$$

Parameters:

Dest Destination, y()
Count Vector length, N

5.10.3.320 void cDSPOp::WinTukey (double *, long)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.321 void cIDSPOp::WinDolphChebyshev (float *, float, long)

Calculate Dolph-Chebyshev window coefficients.

$$w(n) = \frac{1}{N} \left[1 + 2r \sum_{m=1}^M T_{2M} \left(x_0 \cos \frac{\theta_m}{2} \right) \cos m\theta_n \right], -M \leq n \leq M$$

$$x_0 = \cosh \left(\frac{1}{2M} \cosh^{-1} \frac{1}{r} \right)$$

Gamma (r) is sidelobe / mainlobe ratio.

Parameters:

Dest Destination w()

Gamma Gamma (r)

Count Vector length, N

5.10.3.322 void cIDSPOp::WinDolphChebyshev (double *, double, long)**5.10.3.323 void cIDSPOp::Mix (float *, const float *, long) [static]**

Mix two channels interleaved in vector.

Parameters:

Dest Destination

Src Source

Count Vector length (destination)

5.10.3.324 void cIDSPOp::Mix (double *, const double *, long) [static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.325 void cIDSPOp::Mix (float *, const float *, const float *, long) [static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Mix two channels in separate vectors

5.10.3.326 void cIDSPOp::Mix (double *, const double *, const double *, long) [static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.327 void cIDSPOp::Mix (float *, const float *, long, long) [static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Mix n channels interleaved in vector

5.10.3.328 void cIDSPOp::Mix (double *, const double *, long, long) [static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.329 void cIDSPOp::Spatialize (float *, float *, const float *, long)

Spatialize one channel to two channels by inversing others phase.

For playing mono sound on stereo headphones.

Parameters:

Dest1 Destination 1

Dest2 Destination 2

Src Source

Count Vector length

5.10.3.330 void cIDSPOp::Spatialize (double *, double *, const double *, long)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.331 void cIDSPOp::Spatialize (float *, const float *, long)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.332 void cIDSPOp::Spatialize (double *, const double *, long)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.333 void cIDSPOp::Extract (float *, const float *, long, long, long) [static]

Extract channel n from N channel interleaved input.

Parameters:*Dest* Destination*Src* Source*Channel* Channel to extract*ChCount* Channel count*Count* Vector length (source)

5.10.3.334 `void cIDSPOp::Extract (double *, const double *, long, long, long)`
[static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.335 `void cIDSPOp::Pack (float *, const float *, long, long, long)`
[static]

Pack channel n to N channel interleaved output.

Parameters:*Dest* Destination*Src* Source*Channel* Channel to pack*ChCount* Channel count*Count* Vector length (source)

5.10.3.336 `void cIDSPOp::Pack (double *, const double *, long, long, long)`
[static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.337 `long cIDSPOp::ReBuffer (float *, const float *, long, long)`

Rebuffer to different sized buffer.

Returns 0 if there was no complete result block, recall with new source buffer and same destination buffer.

Returns 1 if there was complete result block, recall with new source and new destination buffer next time.

Returns 2 if source buffer is not yet empty, but one result buffer available, recall with same source buffer and new destination buffer next time.

Parameters:

Dest Destination
Src Source
DestCount Destination vector length
SrcCount Source vector length

Returns:

Rebuffering result

5.10.3.338 long cIDSPOp::ReBuffer (double *, const double *, long, long)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.339 double cIDSPOp::DegToRad (double *dSource*) [inline]

Convert degrees to radians.

$$y = \frac{\pi}{180}x$$

Parameters:

Source Degrees, x

Returns:

Radians, y

5.10.3.340 float cIDSPOp::DegToRad (float *fSource*) [inline]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.341 double cIDSPOp::RadToDeg (double *dSource*) [inline]

Convert radians to degrees.

$$y = \frac{180}{\pi}x$$

Parameters:

Source Radians, x
Degrees,y

5.10.3.342 `float cIDSPOp::RadToDeg (float fSource) [inline]`

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.343 `void cIDSPOp::FFTWConvert (stpSCplx, const float *, long)
[static]`

Convert Real-FFTW complex output to our Cplx.

Note:

Destination length is source length / 2 + 1
This is only for FFTW < 3.x

5.10.3.344 `void cIDSPOp::FFTWConvert (stpDCplx, const float *, long)
[static]`

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.345 `void cIDSPOp::FFTWConvert (stpSCplx, const double *, long)
[static]`

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.346 `void cIDSPOp::FFTWConvert (stpDCplx, const double *, long)
[static]`

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.347 `void cIDSPOp::FFTWConvert (float *, const stpSCplx, long)
[static]`

Convert our Cplx to Real-FFTW complex input.

5.10.3.348 `void cIDSPOp::FFTWConvert (float *, const stpDCplx, long)
[static]`

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.349 void cDSPOp::FFTWConvert (double *, const *stpSCplx*, long)
[static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.350 void cDSPOp::FFTWConvert (double *, const *stpDCplx*, long)
[static]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.351 void cDSPOp::FIRAllocate (const float *, long)

Allocate FIR filter.

Parameters:

Coeff Coefficients

Count Vector length

5.10.3.352 void cDSPOp::FIRAllocate (const double *, long)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.353 void cDSPOp::FIRFilter (float *, long)

Filter using FIR filter.

Parameters:

Vect Source & destination

Count Vector length

5.10.3.354 void cDSPOp::FIRFilter (double *, long)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.355 void cDSPOp::FIRFilter (float *, const float *, long)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Parameters:*Dest* Destination*Src* Source*Count* Vector length**5.10.3.356 void cIDSPOp::FIRFilter (double *, const double *, long)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.357 void cIDSPOp::FIRFilterF (float *, float *, long)

Filter using FIR filter (fast version).

Note:

New data at source[FIRlength], must have FIRlength size scratch pad at start of source vector.

Parameters:*Dest* Destination*Src* Source*Count* Vector length**5.10.3.358 void cIDSPOp::FIRFilterF (double *, double *, long)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.359 void cIDSPOp::FIRFree ()

Free FIR filter.

5.10.3.360 void cIDSPOp::IIRInitialize (const float *)

Initialize IIR filter.

Data format is: [0..2] = b[0..2] [3..4] = a[0..1]

Parameters:*Coeffs* Coefficient vector of length 5

5.10.3.361 void cDSPOp::IIRInitialize (const double *)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.362 void cDSPOp::IIRFilter (float *, long)

Filter using IIR filter.

$$H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 - a_0 z^{-1} - a_1 z^{-2}}$$

$$y(n) = \sum_{i=1}^M a_i y(n-i) + \sum_{j=0}^N b_j x(n-j)$$

Parameters:

Vect Source & destination

Count Vector length

5.10.3.363 void cDSPOp::IIRFilter (double *, long)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.364 void cDSPOp::IIRFilter (float *, const float *, long)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Parameters:

Dest Destination

Src Source

Count Vector length

5.10.3.365 void cDSPOp::IIRFilter (double *, const double *, long)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.366 void cDSPOp::IIRClear ()

Clear IIR filter feedback chain.

5.10.3.367 void cIDSPOp::FFTInitialize (long, bool)

Initialize FFT (and other transforms).

Parameters:

Size Transform size

Real Real transform?

5.10.3.368 void cIDSPOp::FFTUninitialize ()

Uninitialize FFT.

5.10.3.369 void cIDSPOp::FFTi (stpSCplx, float *)

FFT half in-place, source vector is modified.

Note:

Output is FFT size / 2 + 1

Parameters:

Dest Destination

Src Source

5.10.3.370 void cIDSPOp::FFTi (stpDCplx, double *)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.371 void cIDSPOp::FFTo (stpSCplx, const float *)

FFT out-of-place.

Note:

Output is FFT size / 2 + 1 for real input

Parameters:

Dest Destination

Src Source

5.10.3.372 void cIDSPOp::FFTo (stpDCplx, const double *)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.373 void cDSPOp::FFTo (stpSCplx, const stpSCplx)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.374 void cDSPOp::FFTo (stpDCplx, const stpDCplx)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.375 void cDSPOp::IFFTo (float *, const stpSCplx)

IFFT out-of-place.

Note:

Input is FFT size / 2 + 1 for real output

Parameters:

Dest Destination

Src Source

5.10.3.376 void cDSPOp::IFFTo (double *, const stpDCplx)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.377 void cDSPOp::IFFTo (stpSCplx, const stpSCplx)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.3.378 void cDSPOp::IFFTo (stpDCplx, const stpDCplx)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.10.4 Member Data Documentation

5.10.4.1 long [cDSPOp::lPrevSrcCount](#) [private]

5.10.4.2 long [cDSPOp::lPrevDestCount](#) [private]

5.10.4.3 float [cDSPOp::fPI](#) [private]

5.10.4.4 double [cDSPOp::dPI](#) [private]

5.10.4.5 int [cDSPOp::iFIRDlyIdx](#) [private]

5.10.4.6 long [cDSPOp::lFIRLength](#) [private]

5.10.4.7 [cDSPAlloc cDSPOp::FIRCoeff](#) [private]

5.10.4.8 [cDSPAlloc cDSPOp::FIRBuf](#) [private]

5.10.4.9 [cDSPAlloc cDSPOp::FIRWork](#) [private]

5.10.4.10 float [cDSPOp::fpIIR_C\[5\]](#) [private]

5.10.4.11 float [cDSPOp::fpIIR_X\[3\]](#) [private]

5.10.4.12 float [cDSPOp::fpIIR_Y\[2\]](#) [private]

5.10.4.13 double [cDSPOp::dpIIR_C\[5\]](#) [private]

5.10.4.14 double [cDSPOp::dpIIR_X\[3\]](#) [private]

5.10.4.15 double [cDSPOp::dpIIR_Y\[2\]](#) [private]

5.10.4.16 bool [cDSPOp::bFFTInitialized](#) [private]

5.10.4.17 bool [cDSPOp::bRealTransform](#) [private]

5.10.4.18 long [cDSPOp::lFFTLLength](#) [private]

5.10.4.19 float [cDSPOp::fFFTSscale](#) [private]

5.10.4.20 double [cDSPOp::dFFTSscale](#) [private]

5.10.4.21 long* [cDSPOp::lpSBitRevWork](#) [private]

5.10.4.22 long* [cDSPOp::lpDBitRevWork](#) [private]

5.10.4.23 float* [cDSPOp::fpCosSinTable](#) [private]

5.10.4.24 double* [cDSPOp::dpCosSinTable](#) [private]

5.10.4.25 void* [cDSPOp::vpSTfrm](#) [private]

5.10.4.26 void* [cDSPOp::vpDTfrm](#) [private]

5.10.4.27 [cDSPAlloc cDSPOp::SBitRevWork](#) [private]

5.10.4.28 [cDSPAlloc cDSPOp::DBitRevWork](#) [private]

5.10.4.29 [cDSPAlloc cDSPOp::SCosSinTable](#) [private]

5.11 `clDSPVector< TDSPVector_t >` Class Template Reference

`#include <DSPVector.hh>`

Inherits [clReBufferT< TDSPVector_t >](#).

Public Member Functions

- [clDSPVector](#) ()
- [clDSPVector](#) (const [clDSPVector](#) &CopySrc)
- [clDSPVector](#) (long lNewSize)
- [clDSPVector](#) (const TDSPVector_t *fpSrcData, long lSrcCount)
- [~clDSPVector](#) ()
- TDSPVector_t * [Ptr](#) ()
- [clDSPVector operator+](#) (const TDSPVector_t &Src)
- [clDSPVector operator+](#) ([clDSPVector](#) &Src)
- [clDSPVector operator-](#) (const TDSPVector_t &Src)
- [clDSPVector operator-](#) ([clDSPVector](#) &Src)
- [clDSPVector operator *](#) (const TDSPVector_t &Src)
- [clDSPVector operator *](#) ([clDSPVector](#) &Src)
- [clDSPVector operator/](#) (const TDSPVector_t &Src)
- [clDSPVector operator/](#) ([clDSPVector](#) &Src)
- [clDSPVector & operator+=](#) (const TDSPVector_t &Src)
- [clDSPVector & operator+=](#) ([clDSPVector](#) &Src)
- [clDSPVector & operator-=](#) (const TDSPVector_t &Src)
- [clDSPVector & operator-=](#) ([clDSPVector](#) &Src)
- [clDSPVector & operator *=](#) (const TDSPVector_t &Src)
- [clDSPVector & operator *=](#) ([clDSPVector](#) &Src)
- [clDSPVector & operator/=](#) (const TDSPVector_t &Src)
- [clDSPVector & operator/=](#) ([clDSPVector](#) &Src)
- [clDSPVector & Zero](#) ()
- [clDSPVector & Set](#) (const TDSPVector_t &Src)
- [clDSPVector & Set](#) (const TDSPVector_t *Src, long lSrcCount)
- [clDSPVector & Set](#) (const TDSPVector_t &Src, long lStartIdx, long lSetCount)
- [clDSPVector & Clip](#) (const TDSPVector_t &Src)
- [clDSPVector & Clip](#) (const TDSPVector_t &Src1, const TDSPVector_t &Src2)
- [clDSPVector & ClipZero](#) ()
- [clDSPVector & MulC](#) ([clDSPVector](#) &Src)
- [clDSPVector & MulC](#) ([clDSPVector](#) &Src1, [clDSPVector](#) &Src2)
- [clDSPVector & Div1x](#) ()
- [clDSPVector & MulAdd](#) (TDSPVector_t fMul, TDSPVector_t fAdd)
- [clDSPVector & Square](#) ()
- [clDSPVector & Abs](#) ()
- [clDSPVector & Sqrt](#) ()
- [clDSPVector & Negate](#) ()

- [cIDSPVector & Normalize \(\)](#)
- [cIDSPVector & Reverse \(\)](#)
- [cIDSPVector & Reverse \(cIDSPVector &Src\)](#)
- [cIDSPVector & Scale \(\)](#)
- [cIDSPVector & Scale \(cIDSPVector &Src\)](#)
- [cIDSPVector & Scale01 \(\)](#)
- [cIDSPVector & Scale01 \(cIDSPVector &Src\)](#)
- [cIDSPVector & Sort \(\)](#)
- [cIDSPVector & Sort \(cIDSPVector &Src\)](#)
- [TDSPVector_t Sum \(\)](#)
- [TDSPVector_t Conv \(cIDSPVector &Src\)](#)
- [cIDSPVector & Conv \(cIDSPVector &Src1, cIDSPVector &Src2\)](#)
- [TDSPVector_t Corr \(cIDSPVector &Src\)](#)
- [cIDSPVector & Corr \(cIDSPVector &Src1, cIDSPVector &Src2\)](#)
- [TDSPVector_t AutoCorr \(\)](#)
- [cIDSPVector & AutoCorr \(cIDSPVector &Src\)](#)
- [TDSPVector_t CrossCorr \(cIDSPVector &Src\)](#)
- [TDSPVector_t CrossCorr \(cIDSPVector &Src, long lDelay\)](#)
- [cIDSPVector & CrossCorr \(cIDSPVector &Src1, cIDSPVector &Src2, long *lpDelays, long lDelayCount\)](#)
- [TDSPVector_t DotProduct \(cIDSPVector &Src\)](#)
- [TDSPVector_t Mean \(\)](#)
- [TDSPVector_t Median \(\)](#)
- [TDSPVector_t Energy \(\)](#)
- [TDSPVector_t RMS \(\)](#)
- [TDSPVector_t PeakLevel \(\)](#)
- [void MinMax \(TDSPVector_t &fMin, TDSPVector_t &fMax\)](#)
- [void StdDev \(TDSPVector_t &fStdDev, TDSPVector_t &fMean\)](#)
- [void Variance \(TDSPVector_t &fVariance, TDSPVector_t &fMean\)](#)
- [cIDSPVector & Convert \(const unsigned char *ucpSrc, long lSrcCount\)](#)
- [cIDSPVector & Convert \(const signed short *ipSrc, long lSrcCount, bool b12bit=false\)](#)
- [cIDSPVector & Convert \(const signed int *ipSrc, long lSrcCount, bool b24bit=false\)](#)
- [cIDSPVector & Convert \(const float *fpSrc, long lSrcCount\)](#)
- [cIDSPVector & Convert \(const double *dpSrc, long lSrcCount\)](#)
- [void Convert \(unsigned char *ucpDest\)](#)
- [void Convert \(signed short *ipDest, bool b12bit=false\)](#)
- [void Convert \(signed int *ipDest, bool b24bit=false\)](#)
- [void Convert \(float *fpDest\)](#)
- [void Convert \(double *dpDest\)](#)
- [cIDSPVector & CartToPolar \(\)](#)
- [cIDSPVector & CartToPolar \(cIDSPVector &Cart\)](#)
- [void CartToPolar \(cIDSPVector< float > &Magn, cIDSPVector< float > &Phase\)](#)
- [void CartToPolar \(cIDSPVector< double > &Magn, cIDSPVector< double > &Phase\)](#)

- `clDSPVector` & `PolarToCart` ()
- `clDSPVector` & `PolarToCart` (`clDSPVector` &Polar)
- void `PolarToCart` (`clDSPVector`< float > &Real, `clDSPVector`< float > &Imag)
- void `PolarToCart` (`clDSPVector`< double > &Real, `clDSPVector`< double > &Imag)
- `clDSPVector` & `Magnitude` (`clDSPVector`< `stSCplx` > &Src)
- `clDSPVector` & `Magnitude` (`clDSPVector`< `stDCplx` > &Src)
- `clDSPVector` & `Power` (`clDSPVector`< `stSCplx` > &Src)
- `clDSPVector` & `Power` (`clDSPVector`< `stDCplx` > &Src)
- `clDSPVector` & `Phase` (`clDSPVector`< `stSCplx` > &Src)
- `clDSPVector` & `Phase` (`clDSPVector`< `stDCplx` > &Src)
- void `PowerPhase` (`clDSPVector`< float > &Power, `clDSPVector`< float > &Phase)
- void `PowerPhase` (`clDSPVector`< double > &Power, `clDSPVector`< double > &Phase)
- `clDSPVector` & `Decimate` (long lFactor)
- `clDSPVector` & `Decimate` (`clDSPVector` &Src, long lFactor)
- `clDSPVector` & `DecimateAvg` (`clDSPVector` &Src, long lFactor)
- `clDSPVector` & `Interpolate` (`clDSPVector` &Src, long lFactor)
- `clDSPVector` & `InterpolateAvg` (`clDSPVector` &Src, long lFactor)
- `clDSPVector` & `Resample` (`clDSPVector` &Src)
- `clDSPVector` & `ResampleAvg` (`clDSPVector` &Src)
- `clDSPVector` & `WinBartlett` (long lWinSize)
- `clDSPVector` & `WinBlackman` (long lWinSize)
- `clDSPVector` & `WinBlackman` (long lWinSize, `TDSPVector_t` fAlpha)
- `clDSPVector` & `WinBlackmanHarris` (long lWinSize)
- `clDSPVector` & `WinCosTapered` (long lWinSize)
- `clDSPVector` & `WinExactBlackman` (long lWinSize)
- `clDSPVector` & `WinExp` (`TDSPVector_t` fZ, long lWinSize)
- `clDSPVector` & `WinFlatTop` (long lWinSize)
- `clDSPVector` & `WinGenericCos` (`clDSPVector` &Coeffs, long lWinSize)
- `clDSPVector` & `WinHamming` (long lWinSize)
- `clDSPVector` & `WinHanning` (long lWinSize)
- `clDSPVector` & `WinKaiser` (`TDSPVector_t` fBeta, long lWinSize)
- `clDSPVector` & `WinKaiserBessel` (`TDSPVector_t` fAlpha, long lWinSize)
- `clDSPVector` & `WinTukey` (long lWinSize)
- `clDSPVector` & `WinDolphChebyshev` (`TDSPVector_t` fGamma, long lWinSize)
- `clDSPVector` & `Mix` (`clDSPVector` &Src, long lChCount)
- `clDSPVector` & `Extract` (`clDSPVector` &Src, long lCh, long lChCount)
- `clDSPVector` & `Pack` (`clDSPVector` &Src, long lCh, long lChCount)
- `clDSPVector` & `FFTWConvert` (`clDSPVector` &Src)

Vector must be sized before calling this!

- void `FIRAllocate` (`clDSPVector` &Src)
- `clDSPVector` & `FIRFilter` ()
- `clDSPVector` & `FIRFilter` (`clDSPVector` &Src)
- void `FIRFree` ()

- void [FFTInitialize](#) (long lWinSize, bool bReal)
- [clDSPVector](#) & [FFT](#) ([clDSPVector](#)< float > &Src)
- [clDSPVector](#) & [FFT](#) ([clDSPVector](#)< double > &Src)
- [clDSPVector](#) & [FFT](#) ([clDSPVector](#)< [stSCplx](#) > &Src)
- [clDSPVector](#) & [FFT](#) ([clDSPVector](#)< [stDCplx](#) > &Src)
- [clDSPVector](#) & [IFFT](#) ([clDSPVector](#)< [stSCplx](#) > &Src)
- [clDSPVector](#) & [IFFT](#) ([clDSPVector](#)< [stDCplx](#) > &Src)
- void [FFTUninitialize](#) ()

Private Attributes

- long [lFFTSIZE](#)
- [clDSPOp](#) DSP


```
template<class TDSPVector_t> class cIDSPVector< TDSPVector_t >
```

5.11.1 Constructor & Destructor Documentation

5.11.1.1 `template<class TDSPVector_t> cIDSPVector< TDSPVector_t >::cIDSPVector() [inline]`

5.11.1.2 `template<class TDSPVector_t> cIDSPVector< TDSPVector_t >::cIDSPVector(const cIDSPVector< TDSPVector_t > & CopySrc) [inline]`

5.11.1.3 `template<class TDSPVector_t> cIDSPVector< TDSPVector_t >::cIDSPVector(long lNewSize) [inline]`

5.11.1.4 `template<class TDSPVector_t> cIDSPVector< TDSPVector_t >::cIDSPVector(const TDSPVector_t * fpSrcData, long lSrcCount) [inline]`

5.11.1.5 `template<class TDSPVector_t> cIDSPVector< TDSPVector_t >::~~cIDSPVector() [inline]`

5.11.2 Member Function Documentation

5.11.2.1 `template<class TDSPVector_t> TDSPVector_t* cIDSPVector< TDSPVector_t >::Ptr() [inline]`

5.11.2.2 `template<class TDSPVector_t> cIDSPVector cIDSPVector< TDSPVector_t >::operator+ (const TDSPVector_t & Src) [inline]`

5.11.2.3 `template<class TDSPVector_t> cIDSPVector cIDSPVector< TDSPVector_t >::operator+ (cIDSPVector< TDSPVector_t > & Src) [inline]`

5.11.2.4 `template<class TDSPVector_t> cIDSPVector cIDSPVector< TDSPVector_t >::operator- (const TDSPVector_t & Src) [inline]`

5.11.2.5 `template<class TDSPVector_t> cIDSPVector cIDSPVector< TDSPVector_t >::operator- (cIDSPVector< TDSPVector_t > & Src) [inline]`

5.11.2.6 `template<class TDSPVector_t> cIDSPVector cIDSPVector< TDSPVector_t >::operator * (const TDSPVector_t & Src) [inline]`

5.11.2.7 `template<class TDSPVector_t> cIDSPVector cIDSPVector< TDSPVector_t >::operator * (cIDSPVector< TDSPVector_t > & Src) [inline]`

5.11.2.8 `template<class TDSPVector_t> cIDSPVector cIDSPVector< TDSPVector_t >::operator/ (const TDSPVector_t & Src) [inline]`

5.11.2.9 `template<class TDSPVector_t> cIDSPVector cIDSPVector< TDSPVector_t >::operator/ (cIDSPVector< TDSPVector_t > & Src) [inline]`

5.11.2.10 `template<class TDSPVector_t> cIDSPVector& cIDSPVector< TDSPVector_t >::operator+= (const TDSPVector_t & Src) [inline]`

5.11.2.11 `template<class TDSPVector_t> cIDSPVector& cIDSPVector<`

- 5.11.2.113 `template<class TDSPVector_t> void cIDSPVector< TDSPVector_t >::FIRAllocate (cIDSPVector< TDSPVector_t > & Src)`
[inline]
- 5.11.2.114 `template<class TDSPVector_t> cIDSPVector& cIDSPVector< TDSPVector_t >::FIRFilter ()` [inline]
- 5.11.2.115 `template<class TDSPVector_t> cIDSPVector& cIDSPVector< TDSPVector_t >::FIRFilter (cIDSPVector< TDSPVector_t > & Src)`
[inline]
- 5.11.2.116 `template<class TDSPVector_t> void cIDSPVector< TDSPVector_t >::FIRFree ()` [inline]
- 5.11.2.117 `template<class TDSPVector_t> void cIDSPVector< TDSPVector_t >::FFTInitialize (long IWinSize, bool bReal)` [inline]
- 5.11.2.118 `template<class TDSPVector_t> cIDSPVector& cIDSPVector< TDSPVector_t >::FFT (cIDSPVector< float > & Src)` [inline]
- 5.11.2.119 `template<class TDSPVector_t> cIDSPVector& cIDSPVector< TDSPVector_t >::FFT (cIDSPVector< double > & Src)` [inline]
- 5.11.2.120 `template<class TDSPVector_t> cIDSPVector& cIDSPVector< TDSPVector_t >::FFT (cIDSPVector< stSCplx > & Src)`
[inline]
- 5.11.2.121 `template<class TDSPVector_t> cIDSPVector& cIDSPVector< TDSPVector_t >::FFT (cIDSPVector< stDCplx > & Src)`
[inline]
- 5.11.2.122 `template<class TDSPVector_t> cIDSPVector& cIDSPVector< TDSPVector_t >::IFFT (cIDSPVector< stSCplx > & Src)`
[inline]
- 5.11.2.123 `template<class TDSPVector_t> cIDSPVector& cIDSPVector< TDSPVector_t >::IFFT (cIDSPVector< stDCplx > & Src)`
[inline]
- 5.11.2.124 `template<class TDSPVector_t> void cIDSPVector< TDSPVector_t >::FFTUninitialize ()` [inline]

5.11.3 Member Data Documentation

- 5.11.3.1 `template<class TDSPVector_t> long cIDSPVector< TDSPVector_t >::IFFTSize` [private]
- 5.11.3.2 `template<class TDSPVector_t> cIDSPOp cIDSPVector< TDSPVector_t >::DSP` [private]

5.12 `clDynThreads< TThreads >` Class Template Reference

`#include <DynThreads.hh>`

Inherits [clDynThreadsBase](#).

Public Types

- typedef void *(TThreads::* [Method_t](#))(void *)
- typedef [clDynThreads::_stParams2](#) stParams2
- typedef [clDynThreads::_stParams2](#) * stpParams2

Public Member Functions

- [clDynThreads](#) (TThreads &KlassInst)
- int [Create](#) ([Method_t](#) Method, void *vpParam, bool bDetached=false)
- virtual void * [InternalCaller](#) (void *vpParam)

Public Attributes

- TThreads * [Klass](#)

```
template<class TThreads> class clDynThreads< TThreads >
```

5.12.1 Member Typedef Documentation

5.12.1.1 `template<class TThreads> typedef void*(TThreads::*
clDynThreads< TThreads >::Method_t)(void *)`

5.12.1.2 `template<class TThreads> typedef struct clDynThreads::_stParams2
clDynThreads< TThreads >::stParams2`

5.12.1.3 `template<class TThreads> typedef struct clDynThreads::_stParams2
* clDynThreads< TThreads >::stpParams2`

5.12.2 Constructor & Destructor Documentation

5.12.2.1 `template<class TThreads> clDynThreads< TThreads
>::clDynThreads (TThreads & KlassInst) [inline]`

5.12.3 Member Function Documentation

5.12.3.1 `template<class TThreads> int clDynThreads< TThreads >::Create
(Method_t Method, void * vpParam, bool bDetached = false)
[inline]`

5.12.3.2 `template<class TThreads> virtual void* clDynThreads< TThreads
>::InternalCaller (void * vpParam) [inline, virtual]`

Implements [clDynThreadsBase](#).

5.12.4 Member Data Documentation

5.12.4.1 `template<class TThreads> TThreads* clDynThreads< TThreads
>::Klass`

5.13 `clDynThreads< TThreads >::_stParams2` Struct Reference

```
#include <DynThreads.hh>
```

Public Attributes

- [Method_t Method](#)
- `void * vpParam`

```
template<class TThreads> struct clDynThreads< TThreads >::_stParams2
```

5.13.1 Member Data Documentation

5.13.1.1 `template<class TThreads> Method_t clDynThreads< TThreads >::_stParams2::Method`

5.13.1.2 `template<class TThreads> void* clDynThreads< TThreads >::_stParams2::vpParam`

5.14 clDynThreadsBase Class Reference

```
#include <DynThreads.hh>
```

Inherited by [clDynThreads< TThreads >](#).

Public Types

- typedef [clDynThreadsBase::_stParams](#) stParams
- typedef [clDynThreadsBase::_stParams](#) * stpParams

Public Member Functions

- [clDynThreadsBase](#) ()
- virtual [~clDynThreadsBase](#) ()
- int [Create](#) (void *, bool)
- void * [Wait](#) (int)
- pthread_t [Self](#) ()
- bool [SetSched](#) (pthread_t, int, int)
- virtual void * [InternalCaller](#) (void *)=0

Private Attributes

- int [iThreadCount](#)
- [ThreadMap_t](#) mapThreads
- [clMutex](#) MtxBase

5.14.1 Member Typedef Documentation

5.14.1.1 typedef struct [clDynThreadsBase::_stParams](#)
[clDynThreadsBase::stParams](#)

5.14.1.2 typedef struct [clDynThreadsBase::_stParams](#) *
[clDynThreadsBase::stpParams](#)

5.14.2 Constructor & Destructor Documentation

5.14.2.1 [clDynThreadsBase::clDynThreadsBase](#) ()

5.14.2.2 [clDynThreadsBase::~~clDynThreadsBase](#) () [virtual]

5.14.3 Member Function Documentation

5.14.3.1 int [clDynThreadsBase::Create](#) (void *, bool)

5.14.3.2 void * [clDynThreadsBase::Wait](#) (int)

5.14.3.3 pthread_t [clDynThreadsBase::Self](#) () [inline]

5.14.3.4 bool [clDynThreadsBase::SetSched](#) (pthread_t, int, int)

5.14.3.5 virtual void* [clDynThreadsBase::InternalCaller](#) (void *) [pure virtual]

Implemented in [clDynThreads< TThreads >](#).

5.14.4 Member Data Documentation

5.14.4.1 int [clDynThreadsBase::iThreadCount](#) [private]

5.14.4.2 [ThreadMap_t](#) [clDynThreadsBase::mapThreads](#) [private]

5.14.4.3 [clMutex](#) [clDynThreadsBase::MtxBase](#) [private]

5.15 clDynThreadsBase::_stParams Struct Reference

```
#include <DynThreads.hh>
```

Public Attributes

- [clDynThreadsBase * Klass](#)
- [void * vpParam](#)

5.15.1 Member Data Documentation

5.15.1.1 [clDynThreadsBase*](#) [clDynThreadsBase::_stParams::Klass](#)

5.15.1.2 [void*](#) [clDynThreadsBase::_stParams::vpParam](#)

5.16 clException Class Reference

```
#include <Exception.hh>
```

Public Member Functions

- [clException](#) () throw ()
- [clException](#) (int iErrorCode) throw ()
- [clException](#) (const std::string &strErrorMsg) throw ()
- [clException](#) (const std::string &strErrorMsg, int iErrorCode) throw ()
- [clException](#) (const char *cpErrorMsg) throw ()
- [clException](#) (const char *cpErrorMsg, int iErrorCode) throw ()
- virtual [~clException](#) () throw ()
- [operator int](#) () const throw ()
- [operator std::string](#) () const throw ()
- virtual const char * [what](#) () const throw ()

Private Attributes

- int [iCode](#)
- std::string [strMsg](#)

5.16.1 Constructor & Destructor Documentation

5.16.1.1 `clException::clException () throw ()` [inline]

5.16.1.2 `clException::clException (int iErrorCode) throw ()` [inline]

5.16.1.3 `clException::clException (const std::string & strErrorMsg) throw ()`
[inline]

5.16.1.4 `clException::clException (const std::string & strErrorMsg, int
iErrorCode) throw ()` [inline]

5.16.1.5 `clException::clException (const char * cpErrorMsg) throw ()`
[inline]

5.16.1.6 `clException::clException (const char * cpErrorMsg, int iErrorCode)
throw ()` [inline]

5.16.1.7 `virtual clException::~clException () throw ()` [inline, virtual]

5.16.2 Member Function Documentation

5.16.2.1 `clException::operator int () const throw ()` [inline]

5.16.2.2 `clException::operator std::string () const throw ()` [inline]

5.16.2.3 `virtual const char* clException::what () const throw ()` [inline,
virtual]

5.16.3 Member Data Documentation

5.16.3.1 `int clException::iCode` [private]

5.16.3.2 `std::string clException::strMsg` [private]

5.17 cIFFTDecimator Class Reference

FFT decimation filter class implementation.

```
#include <IFFTDecimator.hh>
```

Inherits [cIFFTMultiRate](#).

Public Member Functions

- [cIFFTDecimator](#) ()
- [~cIFFTDecimator](#) ()
- void [Uninitialize](#) ()
Uninitialize decimator.
- void [Put](#) (const float *, long)
Feed data into decimator.
- void [Put](#) (const double *, long)
- bool [Get](#) (float *, long)
Get data from decimator.
- bool [Get](#) (double *, long)

Private Attributes

- [cIDSPAlloc](#) DecBuf
- [cIDSPOp](#) DSP

5.17.1 Detailed Description

FFT decimation filter class implementation.

Data is filtered using FFT filter and then decimated.

5.17.2 Constructor & Destructor Documentation

5.17.2.1 cIFFTDecimator::cIFFTDecimator ()

5.17.2.2 cIFFTDecimator::~~cIFFTDecimator ()

5.17.3 Member Function Documentation

5.17.3.1 void cIFFTDecimator::Uninitialize ()

Uninitialize decimator.

Reimplemented from [cIFFTMultiRate](#).

5.17.3.2 void cIFFTDecimator::Put (const float *, long)

Feed data into decimator.

Parameters:

SrcData Source data

SrcCount Source data count

5.17.3.3 void cIFFTDecimator::Put (const double *, long)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.17.3.4 bool cIFFTDecimator::Get (float *, long)

Get data from decimator.

Returns false if there's not enough data feeded into the decimator.

Parameters:

DestData Destination buffer

DestCount Number of samples

Returns:

Data was available?

5.17.3.5 bool cIFFTDecimator::Get (double *, long)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.17.4 Member Data Documentation

5.17.4.1 cDSPAlloc cIFFTDecimator::DecBuf [private]

5.17.4.2 cDSPOp cIFFTDecimator::DSP [private]

5.18 cIFFTInterpolator Class Reference

FFT interpolation filter class implementation.

```
#include <IFFTInterpolator.hh>
```

Inherits [cIFFTMultiRate](#).

Public Member Functions

- [cIFFTInterpolator \(\)](#)
- [~cIFFTInterpolator \(\)](#)
- void [Uninitialize \(\)](#)
Uninitialize interpolator.
- void [Put](#) (const float *, long)
Feed data into interpolator.
- void [Put](#) (const double *, long)
- bool [Get](#) (float *, long)
Get data from interpolator.
- bool [Get](#) (double *, long)

Private Attributes

- [cIDSPAlloc IntBuf](#)
- [cIDSPOp DSP](#)

5.18.1 Detailed Description

FFT interpolation filter class implementation.

Data is interpolated and then filtered using FFT filter.

5.18.2 Constructor & Destructor Documentation

5.18.2.1 cIFFTInterpolator::cIFFTInterpolator ()

5.18.2.2 cIFFTInterpolator::~~cIFFTInterpolator ()

5.18.3 Member Function Documentation

5.18.3.1 void cIFFTInterpolator::Uninitialize ()

Uninitialize interpolator.

Reimplemented from [cIFFTMultiRate](#).

5.18.3.2 void cIFFTInterpolator::Put (const float *, long)

Feed data into interpolator.

Parameters:

SrcData Source data

SrcCount Source data count

5.18.3.3 void cIFFTInterpolator::Put (const double *, long)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.18.3.4 bool cIFFTInterpolator::Get (float *, long)

Get data from interpolator.

Returns false if there's not enough data feeded into the interpolator.

Parameters:

DestData Destination buffer

DestCount Number of samples

Returns:

Data was available?

5.18.3.5 bool cIFFTInterpolator::Get (double *, long)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.18.4 Member Data Documentation

5.18.4.1 cDSPAlloc cIFFTInterpolator::IntBuf [private]

5.18.4.2 cDSPOp cIFFTInterpolator::DSP [private]

5.19 cIFFTMultiRate Class Reference

Base class for FFT based multirate filters.

```
#include <FFTMultiRate.hh>
```

Inherited by [cIFFTDecimator](#), and [cIFFTInterpolator](#).

Public Member Functions

- [cIFFTMultiRate](#) ()
- [~cIFFTMultiRate](#) ()
- bool [Initialize](#) (long, long, const float *, bool=false)
Initialize filter.
- bool [Initialize](#) (long, long, const double *, bool=false)
- void [Uninitialize](#) ()
Uninitialize filter.

Protected Attributes

- bool [bInitialized](#)
- long [lFactor](#)
Rate change factor.
- long [lFilterSize](#)
Size of filter FFT.
- [cIFilter](#) [Filter](#)

5.19.1 Detailed Description

Base class for FFT based multirate filters.

Data is is filtered using FFT filter and then decimated or interpolated by derived classes.

5.19.2 Constructor & Destructor Documentation

5.19.2.1 cIFFTMultiRate::cIFFTMultiRate ()

5.19.2.2 cIFFTMultiRate::~~cIFFTMultiRate ()

5.19.3 Member Function Documentation

5.19.3.1 bool cIFFTMultiRate::Initialize (long, long, const float *, bool = false)

Initialize filter.

You can re-initialize without uninitializing first.

The NULL pointer is used to select correct overloaded function matching input datatype.

Parameters:

FactorP Rate change factor

FiltSize Filter size, specify ≤ 0 for automatic selection

NullPtr NULL pointer

HighPass High-pass filter?

5.19.3.2 bool cIFFTMultiRate::Initialize (long, long, const double *, bool = false)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.19.3.3 void cIFFTMultiRate::Uninitialize ()

Uninitialize filter.

Reimplemented in [cIFFTDecimator](#), and [cIFFTInterpolator](#).

5.19.4 Member Data Documentation

5.19.4.1 bool cIFFTMultiRate::bInitialized [protected]

5.19.4.2 long cIFFTMultiRate::lFactor [protected]

Rate change factor.

5.19.4.3 long cIFFTMultiRate::lFilterSize [protected]

Size of filter FFT.

5.19.4.4 [clFilter clFFTMultiRate::Filter](#) [protected]

5.20 clFilter Class Reference

Class implementing FFT-based FIR-filter with design functions.

#include <Filter.hh>

Inherits [clDSPOp](#).

Public Member Functions

- [clFilter](#) ()
- [~clFilter](#) ()
- bool [Initialize](#) (long, const float *, float=DSP_FILT_DEF_OVERLAPF, float=DSP_FILT_DEF_BETA, int=FILTER_SMOOTH_DOLPH-CHEBYSHEV)
Initialize filter.
- bool [Initialize](#) (long, const double *, double=DSP_FILT_DEF_OVERLAP, double=DSP_FILT_DEF_BETA, int=FILTER_SMOOTH_DOLPH-CHEBYSHEV)
- bool [InitFloat](#) (long)
- bool [InitDouble](#) (long)
- bool [InitializeLP](#) (float, float, float, float=DSP_FILT_DEF_OVERLAPF)
Initialize low-pass or high-pass filter.
- bool [InitializeLP](#) (double, double, double, double=DSP_FILT_DEF_OVERLAP)
- bool [InitializeHP](#) (float, float, float, float=DSP_FILT_DEF_OVERLAPF)
- bool [InitializeHP](#) (double, double, double, double=DSP_FILT_DEF_OVERLAP)
- void [Uninitialize](#) ()
Uninitialize filter.
- void [SetCoeffs](#) (const float *)
Set coefficients.
- void [SetCoeffs](#) (const double *)
- void [SetCoeffs](#) (const [stpSCplx](#), bool=false)
- void [SetCoeffs](#) (const [stpDCplx](#), bool=false)
- void [GetCoeffs](#) (float *)
Get coefficients.
- void [GetCoeffs](#) (double *)
- void [GetCoeffs](#) ([stpSCplx](#))
- void [GetCoeffs](#) ([stpDCplx](#))
- long [GetDelay](#) ()
Get filter delay in samples.

- void [Put](#) (const float *, long)
Feed data into filter.
- void [Put](#) (const double *, long)
- void [Put](#) (const float *, long, const [stpSCplx](#))
Feed data into filter using complex coefficients.
- void [Put](#) (const double *, long, const [stpDCplx](#))
- bool [Get](#) (float *, long)
Get data from filter.
- bool [Get](#) (double *, long)
- void [DesignLP](#) (float *, bool=false)
Design low-pass filter.
- void [DesignLP](#) (double *, bool=false)
- void [DesignLP](#) (float *, float, bool=false)
- void [DesignLP](#) (double *, double, bool=false)
- void [DesignHP](#) (float *)
Design high-pass filter.
- void [DesignHP](#) (double *)
- void [DesignHP](#) (float *, float)
- void [DesignHP](#) (double *, double)
- void [DesignBP](#) (float *, float *)
Design band-pass filter.
- void [DesignBP](#) (double *, double *)
- void [DesignBP](#) (float *, float *, float)
- void [DesignBP](#) (double *, double *, double)
- void [DesignBR](#) (float *, float *)
Design band-reject filter.
- void [DesignBR](#) (double *, double *)
- void [DesignBR](#) (float *, float *, float)
- void [DesignBR](#) (double *, double *, double)

Private Member Functions

- void [InitCoeffsS](#) ()
Initialize complex coefficients.
- void [InitCoeffsD](#) ()
- void [ReadyFilterS](#) ()
Create complex impulse response from specified frequency response.

- void [ReadyFilterD](#) ()
- float [GetKaiserBeta](#) (float fAlpha)
Calculate beta value for Kaiser window.
- double [GetKaiserBeta](#) (double dAlpha)

Private Attributes

- bool [bInitialized](#)
- long [lFFTSize](#)
- long [lHalfSize](#)
- long [lOldSize](#)
- long [lNewSize](#)
- long [lSpectPoints](#)
- [clDSPAlloc](#) CoeffWin
- [clDSPAlloc](#) Prev
- [clDSPAlloc](#) Proc
- [clDSPAlloc](#) CCoeffs
- [clDSPAlloc](#) CProc
- [clReBuffer](#) InBuf
- [clReBuffer](#) OutBuf

5.20.1 Detailed Description

Class implementing FFT-based FIR-filter with design functions.

Filtering is done in complex-plane.

5.20.2 Constructor & Destructor Documentation

5.20.2.1 [clFilter::clFilter](#) ()

5.20.2.2 [clFilter::~~clFilter](#) ()

5.20.3 Member Function Documentation

5.20.3.1 void [clFilter::InitCoeffsS](#) () [private]

Initialize complex coefficients.

$$\left| \begin{array}{l} \Re_C(i) = 1 \\ \Im_C(i) = 0 \end{array} \right.$$

5.20.3.2 void clFilter::InitCoeffsD () [private]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.20.3.3 void clFilter::ReadyFilterS () [private]

Create complex impulse response from specified frequency response.

This is done by doing $FFT^{-1}(X)$, then multiplying it with desired window function and doing $FFT(x)$.

5.20.3.4 void clFilter::ReadyFilterD () [private]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.20.3.5 float clFilter::GetKaiserBeta (float *fAlpha*) [private]

Calculate beta value for Kaiser window.

$$\beta = \begin{cases} 0.1102(\alpha_s - 8.7) & , \alpha_s > 50 \\ 0.5842(\alpha_s - 21)^{0.4} + 0.07886(\alpha_s - 21) & , 21 \leq \alpha_s \leq 50 \\ 0 & , \alpha_s < 21 \end{cases}$$

Parameters:

Alpha Stopband rippleratio in dB

Returns:

Beta for Kaiser window

5.20.3.6 double clFilter::GetKaiserBeta (double *dAlpha*) [private]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.20.3.7 bool clFilter::Initialize (long, const float *, float = DSP_FILT_DEF_OVERLAPF, float = DSP_FILT_DEF_BETAF, int = FILTER_SMOOTH_DOLPH_CHEBYSHEV)

Initialize filter.

Parameters:

WindowSize Number of FFT points used in filter

FiltCoeffs Optional coefficient vector, can be NULL

Overlap Normalized overlap factor

Beta Optional beta value for window

SmoothWindow Type of smoothing window to use

Returns:

Success

5.20.3.8 `bool clFilter::Initialize (long, const double *, double = DSP_FILT_DEF_OVERLAP, double = DSP_FILT_DEF_BETA, int = FILTER_SMOOTH_DOLPH_CHEBYSHEV)`

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.20.3.9 `bool clFilter::InitFloat (long)`

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.20.3.10 `bool clFilter::InitDouble (long)`

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.20.3.11 `bool clFilter::InitializeLP (float, float, float, float = DSP_FILT_DEF_OVERLAPF)`

Initialize low-pass or high-pass filter.

$$N_o = \frac{\alpha_s - 8}{2.285\delta\omega}$$

Parameters:

PassBand Normalized passband

StopBand Normalized stopband

RippleRatio Stopband ripple ratio in dB

Overlap Normalized overlap factor

Returns:

Success

5.20.3.12 **bool cFilter::InitializeLP (double, double, double, double = DSP_FILT_DEF_OVERLAP)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.20.3.13 **bool cFilter::InitializeHP (float, float, float, float = DSP_FILT_DEF_OVERLAPF)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.20.3.14 **bool cFilter::InitializeHP (double, double, double, double = DSP_FILT_DEF_OVERLAP)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.20.3.15 **void cFilter::Uninitialize ()**

Uninitialize filter.

5.20.3.16 **void cFilter::SetCoeffs (const float *)**

Set coefficients.

You can feed the filter with desired frequency response. Length of this dataset is specified window size / 2 + 1. Filter is then created from this coefficient vector.

Parameters:

FiltCoeffs Coefficients for designing filter

Smooth Smooth filter response using smoothing window

5.20.3.17 **void cFilter::SetCoeffs (const double *)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.20.3.18 **void cFilter::SetCoeffs (const stpSCplx, bool = false)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.20.3.19 void clFilter::SetCoeffs (const *stpDCplx*, bool = false)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.20.3.20 void clFilter::GetCoeffs (float *)

Get coefficients.

Get filter's frequency response.

Parameters:

FiltCoeffs Filter coefficients

5.20.3.21 void clFilter::GetCoeffs (double *)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.20.3.22 void clFilter::GetCoeffs (*stpSCplx*)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.20.3.23 void clFilter::GetCoeffs (*stpDCplx*)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.20.3.24 long clFilter::GetDelay ()

Get filter delay in samples.

Delay caused by the filter in number of samples.

Returns:

Delay (in samples)

5.20.3.25 void clFilter::Put (const float *, long)

Feed data into filter.

Filtering is done by first doing $FFT(x)$, then doing complex multiply with complex impulse response, and finally doing $FFT^{-1}(X)$. Windows overlap with specified overlap factor and 1 - overlap factor of window length is used.

Parameters:*SrcData* Source data*SrcCount* Source data count**5.20.3.26 void cFilter::Put (const double *, long)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.20.3.27 void cFilter::Put (const float *, long, const *stpSCplx*)

Feed data into filter using complex coefficients.

Parameters:*SrcData* Source data*SrcCount* Source data count*Coeffs* Complex coefficients**5.20.3.28 void cFilter::Put (const double *, long, const *stpDCplx*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.20.3.29 bool cFilter::Get (float *, long)

Get data from filter.

Parameters:*DestData* Destination buffer*DestCount* Number of samples**Returns:**

Data was available?

5.20.3.30 bool cFilter::Get (double *, long)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.20.3.31 void clFilter::DesignLP (float *, bool = false)

Design low-pass filter.

Parameters:

Corner Corner frequency

SampleRate Sampling frequency

DCBlock Include DC-blocking

5.20.3.32 void clFilter::DesignLP (double *, bool = false)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.20.3.33 void clFilter::DesignLP (float *, float, bool = false)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.20.3.34 void clFilter::DesignLP (double *, double, bool = false)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.20.3.35 void clFilter::DesignHP (float *)

Design high-pass filter.

5.20.3.36 void clFilter::DesignHP (double *)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.20.3.37 void clFilter::DesignHP (float *, float)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.20.3.38 void clFilter::DesignHP (double *, double)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.20.3.39 void clFilter::DesignBP (float *, float *)

Design band-pass filter.

Parameters:

LowCorner Lower corner frequency

HighCorner Higher corner frequency

SampleRate Sampling frequency

5.20.3.40 void clFilter::DesignBP (double *, double *)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.20.3.41 void clFilter::DesignBP (float *, float *, float)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.20.3.42 void clFilter::DesignBP (double *, double *, double)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.20.3.43 void clFilter::DesignBR (float *, float *)

Design band-reject filter.

5.20.3.44 void clFilter::DesignBR (double *, double *)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.20.3.45 void clFilter::DesignBR (float *, float *, float)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.20.3.46 void clFilter::DesignBR (double *, double *, double)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.20.4 Member Data Documentation

- 5.20.4.1 **bool** [clFilter::bInitialized](#) [private]
- 5.20.4.2 **long** [clFilter::lFFTSize](#) [private]
- 5.20.4.3 **long** [clFilter::lHalfSize](#) [private]
- 5.20.4.4 **long** [clFilter::lOldSize](#) [private]
- 5.20.4.5 **long** [clFilter::lNewSize](#) [private]
- 5.20.4.6 **long** [clFilter::lSpectPoints](#) [private]
- 5.20.4.7 **clDSPAlloc** [clFilter::CoeffWin](#) [private]
- 5.20.4.8 **clDSPAlloc** [clFilter::Prev](#) [private]
- 5.20.4.9 **clDSPAlloc** [clFilter::Proc](#) [private]
- 5.20.4.10 **clDSPAlloc** [clFilter::CCoeffs](#) [private]
- 5.20.4.11 **clDSPAlloc** [clFilter::CProc](#) [private]
- 5.20.4.12 **clReBuffer** [clFilter::InBuf](#) [private]
- 5.20.4.13 **clReBuffer** [clFilter::OutBuf](#) [private]

5.21 cFIRDecimator Class Reference

FIR decimation filter class implementation.

```
#include <FIRDecimator.h>
```

Inherits [cFIRMultiRate](#).

Public Member Functions

- [cFIRDecimator](#) ()
- [~cFIRDecimator](#) ()
- void [Uninitialize](#) ()
Uninitialize decimator.
- void [Put](#) (const float *, long)
Feed data into decimator.
- void [Put](#) (const double *, long)
- bool [Get](#) (float *, long)
Get data from decimator.
- bool [Get](#) (double *, long)

Private Attributes

- [cDSPAlloc DecBuf](#)
- [cDSPOp DSP](#)
- [cReBuffer InBuf](#)

5.21.1 Detailed Description

FIR decimation filter class implementation.

Data is filtered using FIR filter and then decimated by factor 2, 4 or 8.

5.21.2 Constructor & Destructor Documentation

5.21.2.1 [cFIRDecimator::cFIRDecimator](#) ()

5.21.2.2 [cFIRDecimator::~~cFIRDecimator](#) ()

5.21.3 Member Function Documentation

5.21.3.1 [void cFIRDecimator::Uninitialize](#) ()

Uninitialize decimator.

Reimplemented from [cFIRMultiRate](#).

5.21.3.2 void cFIRDecimator::Put (const float *, long)

Feed data into decimator.

Parameters:

SrcData Source data

SrcCount Source data count

5.21.3.3 void cFIRDecimator::Put (const double *, long)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.21.3.4 bool cFIRDecimator::Get (float *, long)

Get data from decimator.

Parameters:

DestData Destination buffer

DestCount Number of samples to fetch

Returns:

Success

5.21.3.5 bool cFIRDecimator::Get (double *, long)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.21.4 Member Data Documentation

5.21.4.1 [cDSPAlloc cFIRDecimator::DecBuf](#) [private]

5.21.4.2 [cDSPOp cFIRDecimator::DSP](#) [private]

5.21.4.3 [cReBuffer cFIRDecimator::InBuf](#) [private]

5.22 cFIRInterpolator Class Reference

FIR interpolation filter class implementation.

```
#include <FIRInterpolator.hh>
```

Inherits [cFIRMultiRate](#).

Public Member Functions

- [cFIRInterpolator](#) ()
- [~cFIRInterpolator](#) ()
- void [Uninitialize](#) ()
Uninitialize interpolator.
- void [Put](#) (const float *, long)
Feed data into interpolator.
- void [Put](#) (const double *, long)
- bool [Get](#) (float *, long)
Get data from interpolator.
- bool [Get](#) (double *, long)

Private Attributes

- [cDSPAlloc](#) IntBuf
- [cDSPOp](#) DSP
- [cReBuffer](#) OutBuf

5.22.1 Detailed Description

FIR interpolation filter class implementation.

Data is interpolated by factor 2, 4 or 8 and then filtered using FIR filter.

5.22.2 Constructor & Destructor Documentation

5.22.2.1 [cFIRInterpolator::cFIRInterpolator](#) ()

5.22.2.2 [cFIRInterpolator::~~cFIRInterpolator](#) ()

5.22.3 Member Function Documentation

5.22.3.1 void [cFIRInterpolator::Uninitialize](#) ()

Uninitialize interpolator.

Reimplemented from [cFIRMultiRate](#).

5.22.3.2 void cFIRInterpolator::Put (const float *, long)

Feed data into interpolator.

Parameters:

SrcData Source data

SrcCount Source data count

5.22.3.3 void cFIRInterpolator::Put (const double *, long)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.22.3.4 bool cFIRInterpolator::Get (float *, long)

Get data from interpolator.

Parameters:

DestData Destination buffer

DestCount Number of samples to fetch

Returns:

Success

5.22.3.5 bool cFIRInterpolator::Get (double *, long)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.22.4 Member Data Documentation

5.22.4.1 [cDSPAlloc cFIRInterpolator::IntBuf](#) [private]

5.22.4.2 [cDSPOp cFIRInterpolator::DSP](#) [private]

5.22.4.3 [cReBuffer cFIRInterpolator::OutBuf](#) [private]

5.23 clFIRMultiRate Class Reference

Base class for FIR based multirate filters.

```
#include <FIRMultiRate.hh>
```

Inherited by [clFIRDecimator](#), and [clFIRInterpolator](#).

Public Member Functions

- [clFIRMultiRate](#) ()
- [~clFIRMultiRate](#) ()
- bool [Initialize](#) (long, const float *, bool=false)
Initialize filter, filtering factor must be 2, 3, 4 or 8.
- bool [Initialize](#) (long, const double *, bool=false)
- void [Uninitialize](#) ()
Uninitialize filter.

Protected Attributes

- float [fGain](#)
Filter gain.
- double [dGain](#)
Filter gain.
- long [lFactor](#)
Rate change factor.
- [clDSPOp FIR](#)

5.23.1 Detailed Description

Base class for FIR based multirate filters.

Data is is filtered using FIR filter by factor 2, 3, 4 or 8.

5.23.2 Constructor & Destructor Documentation

5.23.2.1 clFIRMultiRate::clFIRMultiRate ()

5.23.2.2 clFIRMultiRate::~~clFIRMultiRate ()

5.23.3 Member Function Documentation

5.23.3.1 bool clFIRMultiRate::Initialize (long, const float *, bool = false)

Initialize filter, filtering factor must be 2, 3, 4 or 8.

You can re-initialize without uninitializing first.

The NULL pointer is used to select correct overloaded function matching input datatype.

Parameters:

FactorP Rate change factor

NullPtr NULL pointer

HighPass High-pass filter?

5.23.3.2 bool clFIRMultiRate::Initialize (long, const double *, bool = false)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.23.3.3 void clFIRMultiRate::Uninitialize ()

Uninitialize filter.

Reimplemented in [clFIRDecimator](#), and [clFIRInterpolator](#).

5.23.4 Member Data Documentation

5.23.4.1 float clFIRMultiRate::fGain [protected]

Filter gain.

5.23.4.2 double clFIRMultiRate::dGain [protected]

Filter gain.

5.23.4.3 long clFIRMultiRate::lFactor [protected]

Rate change factor.

5.23.4.4 [cIDSPOp cFIRMultiRate::FIR](#) [protected]

5.24 clFlipBand Class Reference

```
#include <FlipBand.hh>
```

Public Member Functions

- [clFlipBand](#) ()
- [~clFlipBand](#) ()
- void [Initialize](#) (long, const float *)
Initialize band flipping.
- void [Initialize](#) (long, const double *)
- void [Uninitialize](#) ()
Uninitialize.
- void [Put](#) (const float *, long)
Put data into flipping buffer.
- void [Put](#) (const double *, long)
- bool [Get](#) (float *, long)
Get data from flipping buffer.
- bool [Get](#) (double *, long)
- void [Clear](#) ()
Clear buffer contents.

Private Attributes

- bool [bInitialized](#)
- long [lBlockSize](#)
- long [lCBlockSize](#)
- [clDSPAlloc Proc](#)
- [clDSPAlloc CProc](#)
- [clDSPOp DSP](#)
- [clReBuffer InBuf](#)
- [clReBuffer OutBuf](#)

5.24.1 Constructor & Destructor Documentation

5.24.1.1 `clFlipBand::clFlipBand ()`

5.24.1.2 `clFlipBand::~~clFlipBand ()`

5.24.2 Member Function Documentation

5.24.2.1 `void clFlipBand::Initialize (long, const float *)`

Initialize band flipping.

Parameters:

TransformSize Size of transform to use for flipping

NullPtr NULL pointer

5.24.2.2 `void clFlipBand::Initialize (long, const double *)`

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.24.2.3 `void clFlipBand::Uninitialize ()`

Uninitialize.

5.24.2.4 `void clFlipBand::Put (const float *, long)`

Put data into flipping buffer.

Parameters:

Src Pointer to source data Count Number of samples to put

5.24.2.5 `void clFlipBand::Put (const double *, long)`

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.24.2.6 `bool clFlipBand::Get (float *, long)`

Get data from flipping buffer.

Parameters:

Dest Pointer to destination buffer

Count Number of samples to get

Returns:

Success?

5.24.2.7 bool cIFlipBand::Get (double *, long)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.24.2.8 void cIFlipBand::Clear ()

Clear buffer contents.

5.24.3 Member Data Documentation

5.24.3.1 bool cIFlipBand::bInitialized [private]

5.24.3.2 long cIFlipBand::lBlockSize [private]

5.24.3.3 long cIFlipBand::lCBlockSize [private]

5.24.3.4 cDSPAlloc cIFlipBand::Proc [private]

5.24.3.5 cDSPAlloc cIFlipBand::CProc [private]

5.24.3.6 cDSPOp cIFlipBand::DSP [private]

5.24.3.7 cReBuffer cIFlipBand::InBuf [private]

5.24.3.8 cReBuffer cIFlipBand::OutBuf [private]

5.25 clHankel Class Reference

Class implementation of (modified) Hankel-transform.

```
#include <Hankel.hh>
```

Public Member Functions

- [clHankel](#) ()
- [~clHankel](#) ()
- void [Initialize](#) (long, const float *)
Initialize Hankel transform.
- void [Initialize](#) (long, const double *)
- void [Uninitialize](#) ()
Uninitialize, destructor also does this.
- void [Process0](#) (float *, const float *)
Process data (0th order).
- void [Process0](#) (double *, const double *)
- void [Process1](#) (float *, const float *)
Process data (1st order).
- void [Process1](#) (double *, const double *)

Private Member Functions

- void [InitAbel](#) (const float *)
Initialize Abel transformer.
- void [InitAbel](#) (const double *)
- void [DoAbel](#) (float *, const float *)
Do Abel transform.
- void [DoAbel](#) (double *, const double *)
- void [UninitAbel](#) ()
Uninitialize Abel transformer.

Private Attributes

- long [lSize](#)
- long [lFFTSIZE](#)
- float [fOutScale0](#)

- float [fOutScale1](#)
- double [dOutScale0](#)
- double [dOutScale1](#)
- [clDSPAlloc A](#)
- [clDSPAlloc B0](#)
- [clDSPAlloc B1](#)
- [clDSPAlloc GX](#)
- [clDSPAlloc GK](#)
- [clDSPOp DSP](#)

5.25.1 Detailed Description

Class implementation of (modified) Hankel-transform.

Author:

Colorado School of Mines
Jussi Laako

Source converted from cwplib (by Dave Hale and Lydia Deng / Colorado School of Mines).

Transform is significantly sped-up by doing Abel-transform followed by inverse-FFT.

References

Hansen, E. W., 1985, Fast Hankel transform algorithm: IEEE Trans. on Acoustics, Speech and Signal Processing, v. ASSP-33, n. 3, p. 666-671. (Beware of several errors in the equations in this paper!)

5.25.2 Constructor & Destructor Documentation

5.25.2.1 `clHankel::clHankel ()`

5.25.2.2 `clHankel::~~clHankel ()`

5.25.3 Member Function Documentation

5.25.3.1 `void clHankel::InitAbel (const float *) [private]`

Initialize Abel transformer.

Parameters:

NullPtr NULL pointer

5.25.3.2 void clHankel::InitAbel (const double *) [private]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.25.3.3 void clHankel::DoAbel (float *, const float *) [private]

Do Abel transform.

Parameters:

Dest Destination

Src Source

5.25.3.4 void clHankel::DoAbel (double *, const double *) [private]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.25.3.5 void clHankel::UninitAbel () [private]

Uninitialize Abel transformer.

5.25.3.6 void clHankel::Initialize (long, const float *)

Initialize Hankel transform.

The NULL pointer is used make C++ compiler select correct datatype for the transform from overloaded method set.

Parameters:

Size Window size

NullPtr NULL pointer

5.25.3.7 void clHankel::Initialize (long, const double *)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.25.3.8 void clHankel::Uninitialize ()

Uninitialize, destructor also does this.

5.25.3.9 void clHankel::Process0 (float *, const float *)

Process data (0th order).

Definition:

$$h_0(k) = \int_0^\infty r j_0(kr) f(r) dr$$

Note:

Input and output is window size / 2 + 1!

Parameters:

Dest Destination

Src Source

5.25.3.10 void clHankel::Process0 (double *, const double *)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.25.3.11 void clHankel::Process1 (float *, const float *)

Process data (1st order).

Definition:

$$h_1(k) = \int_0^\infty r j_1(kr) f(r) dr$$

Note:

Input and output is window size / 2 + 1!

Parameters:

Dest Destination

Src Source

5.25.3.12 void clHankel::Process1 (double *, const double *)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.25.4 Member Data Documentation

- 5.25.4.1 long [clHankel::lSize](#) [private]
- 5.25.4.2 long [clHankel::lFFTSz](#) [private]
- 5.25.4.3 float [clHankel::fOutScale0](#) [private]
- 5.25.4.4 float [clHankel::fOutScale1](#) [private]
- 5.25.4.5 double [clHankel::dOutScale0](#) [private]
- 5.25.4.6 double [clHankel::dOutScale1](#) [private]
- 5.25.4.7 [clDSPAlloc clHankel::A](#) [private]
- 5.25.4.8 [clDSPAlloc clHankel::B0](#) [private]
- 5.25.4.9 [clDSPAlloc clHankel::B1](#) [private]
- 5.25.4.10 [clDSPAlloc clHankel::GX](#) [private]
- 5.25.4.11 [clDSPAlloc clHankel::GK](#) [private]
- 5.25.4.12 [clDSPOp clHankel::DSP](#) [private]

5.26 cIIRCascade Class Reference

Class to handle cascaded IIR stages as one filter.

```
#include <IIRCascade.h>
```

Inherited by [cIIRMultiRate](#)[protected].

Public Member Functions

- [cIIRCascade](#) ()
- [~cIIRCascade](#) ()
- bool [Initialize](#) (const float[][5], long)
Initialize filter.
- bool [Initialize](#) (const double[][5], long)
- void [Uninitialize](#) ()
Uninitialize filter.
- void [Process](#) (float *, long)
Filter data in-place.
- void [Process](#) (double *, long)
- void [Process](#) (float *, const float *, long)
Filter data out-of-place.
- void [Process](#) (double *, const double *, long)
- void [Clear](#) ()
Clear filter feedback chain.

Protected Attributes

- long [lStages](#)
- [cIDSPOp](#) * [IIR](#)

Private Attributes

- bool [bInitialized](#)

5.26.1 Detailed Description

Class to handle cascaded IIR stages as one filter.

5.26.2 Constructor & Destructor Documentation

5.26.2.1 `cliIRCascade::cliIRCascade ()`

5.26.2.2 `cliIRCascade::~~cliIRCascade ()`

5.26.3 Member Function Documentation

5.26.3.1 `bool cliIRCascade::Initialize (const float[][5], long)`

Initialize filter.

Note:

You can re-initialize without uninitializing first.

Parameters:

Coeffs Array of arrays of 5 coefficient bi-quad stages

StageCount Number of cascaded bi-quad filter stages

5.26.3.2 `bool cliIRCascade::Initialize (const double[][5], long)`

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.26.3.3 `void cliIRCascade::Uninitialize ()`

Uninitialize filter.

Reimplemented in [cliIRDecimator](#), [cliIRInterpolator](#), and [cliIRMultirate](#).

5.26.3.4 `void cliIRCascade::Process (float *, long)`

Filter data in-place.

Parameters:

Vect Source & destination vector

Count Vector length

5.26.3.5 `void cliIRCascade::Process (double *, long)`

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.26.3.6 void cIIRCascade::Process (float *, const float *, long)

Filter data out-of-place.

Parameters:

Dest Destination vector

Src Source vector

Count Vector length

5.26.3.7 void cIIRCascade::Process (double *, const double *, long)

5.26.3.8 void cIIRCascade::Clear ()

Clear filter feedback chain.

5.26.4 Member Data Documentation

5.26.4.1 bool [cIIRCascade::bInitialized](#) [private]

5.26.4.2 long [cIIRCascade::lStages](#) [protected]

5.26.4.3 [cIDSPOp*](#) [cIIRCascade::IIR](#) [protected]

5.27 cIIRDecimator Class Reference

IIR decimation filter class implementation.

```
#include <IIRDecimator.hh>
```

Inherits [cIIRMultiRate](#).

Public Member Functions

- [cIIRDecimator](#) ()
- [~cIIRDecimator](#) ()
- void [Uninitialize](#) ()
Uninitialize decimator.
- void [Put](#) (const float *, long)
Feed data into decimator.
- void [Put](#) (const double *, long)
- bool [Get](#) (float *, long)
Get data from decimator.
- bool [Get](#) (double *, long)

Private Attributes

- [cIDSPAlloc DecBuf](#)
- [cIDSPOp DSP](#)
- [cReBuffer InBuf](#)

5.27.1 Detailed Description

IIR decimation filter class implementation.

Data is filtered using IIR filter and then decimated by factor 2 or 3.

5.27.2 Constructor & Destructor Documentation

5.27.2.1 [cIIRDecimator::cIIRDecimator](#) ()

5.27.2.2 [cIIRDecimator::~~cIIRDecimator](#) ()

5.27.3 Member Function Documentation

5.27.3.1 void [cIIRDecimator::Uninitialize](#) ()

Uninitialize decimator.

Reimplemented from [cIIRMultiRate](#).

5.27.3.2 void cIIRDecimator::Put (const float *, long)

Feed data into decimator.

Parameters:

SrcData Source data

SrcCount Source data count

5.27.3.3 void cIIRDecimator::Put (const double *, long)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.27.3.4 bool cIIRDecimator::Get (float *, long)

Get data from decimator.

Parameters:

DestData Destination buffer

DestCount Number of samples to fetch

Returns:

Success

5.27.3.5 bool cIIRDecimator::Get (double *, long)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.27.4 Member Data Documentation

5.27.4.1 [cDSPAlloc cIIRDecimator::DecBuf](#) [private]

5.27.4.2 [cDSPOp cIIRDecimator::DSP](#) [private]

5.27.4.3 [cReBuffer cIIRDecimator::InBuf](#) [private]

5.28 cIIRInterpolator Class Reference

IIR interpolation filter class implementation.

```
#include <IIRInterpolator.hh>
```

Inherits [cIIRMultiRate](#).

Public Member Functions

- [cIIRInterpolator](#) ()
- [~cIIRInterpolator](#) ()
- void [Uninitialize](#) ()
Uninitialize interpolator.
- void [Put](#) (const float *, long)
Feed data into interpolator.
- void [Put](#) (const double *, long)
- bool [Get](#) (float *, long)
Get data from interpolator.
- bool [Get](#) (double *, long)

Private Attributes

- [cDSPAlloc](#) IntBuf
- [cDSPOp](#) DSP
- [cReBuffer](#) OutBuf

5.28.1 Detailed Description

IIR interpolation filter class implementation.

Data is interpolated by factor 2 or 3 and then filtered using IIR filter.

5.28.2 Constructor & Destructor Documentation

5.28.2.1 [cIIRInterpolator::cIIRInterpolator](#) ()

5.28.2.2 [cIIRInterpolator::~~cIIRInterpolator](#) ()

5.28.3 Member Function Documentation

5.28.3.1 void [cIIRInterpolator::Uninitialize](#) ()

Uninitialize interpolator.

Reimplemented from [cIIRMultiRate](#).

5.28.3.2 void cIIRInterpolator::Put (const float *, long)

Feed data into interpolator.

Parameters:

SrcData Source data

SrcCount Source data count

5.28.3.3 void cIIRInterpolator::Put (const double *, long)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.28.3.4 bool cIIRInterpolator::Get (float *, long)

Get data from interpolator.

Parameters:

DestData Destination buffer

DestCount Number of samples to fetch

Returns:

Success

5.28.3.5 bool cIIRInterpolator::Get (double *, long)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.28.4 Member Data Documentation

5.28.4.1 [cDSPAlloc cIIRInterpolator::IntBuf](#) [private]

5.28.4.2 [cDSPOp cIIRInterpolator::DSP](#) [private]

5.28.4.3 [cReBuffer cIIRInterpolator::OutBuf](#) [private]

5.29 cIIRMultiRate Class Reference

Base class for IIR based multirate filters.

`#include <IIRMultiRate.hh>`

Inherits [cIIRCascade](#).

Inherited by [cIIRDecimator](#), and [cIIRInterpolator](#).

Public Member Functions

- [cIIRMultiRate](#) ()
- [~cIIRMultiRate](#) ()
- bool [Initialize](#) (long, const float *, bool=false)
Initialize filter, filtering factor must be 2 or 3.
- bool [Initialize](#) (long, const double *, bool=false)
- void [Uninitialize](#) ()
Uninitialize filter.

Protected Attributes

- long [lFactor](#)
Rate change factor.

5.29.1 Detailed Description

Base class for IIR based multirate filters.

Data is is filtered using IIR filter by factor 2 or 3.

5.29.2 Constructor & Destructor Documentation

5.29.2.1 [cIIRMultiRate::cIIRMultiRate](#) ()

5.29.2.2 [cIIRMultiRate::~~cIIRMultiRate](#) ()

5.29.3 Member Function Documentation

5.29.3.1 [bool cIIRMultiRate::Initialize](#) (long, const float *, bool = false)

Initialize filter, filtering factor must be 2 or 3.

You can re-initialize without uninitializing first.

The NULL pointer is used to select correct overloaded function matching input datatype.

Parameters:

FactorP Rate change factor

NullPtr NULL pointer

HighPass High-pass filter?

5.29.3.2 bool cIIRMultiRate::Initialize (long, const double *, bool = false)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.29.3.3 void cIIRMultiRate::Uninitialize ()

Uninitialize filter.

Reimplemented from [cIIRCascade](#).

Reimplemented in [cIIRDecimator](#), and [cIIRInterpolator](#).

5.29.4 Member Data Documentation**5.29.4.1 long cIIRMultiRate::Factor [protected]**

Rate change factor.

5.30 clMutex Class Reference

Class implementation of POSIX mutex semaphore.

```
#include <Mutex.hh>
```

Public Member Functions

- [clMutex](#) ()
Constructor; creates and initializes the mutex.
- [~clMutex](#) ()
Destructor; destroys the mutex.
- bool [Wait](#) ()
Lock mutex.
- bool [Release](#) ()
Unlock mutex.
- bool [TryLock](#) ()
Try locking mutex without blocking the calling process.
- pthread_mutex_t * [GetPtr](#) ()
Return pointer to mutex variable.

Private Attributes

- pthread_mutex_t [pthmMutex](#)

5.30.1 Detailed Description

Class implementation of POSIX mutex semaphore.

5.30.2 Constructor & Destructor Documentation

5.30.2.1 clMutex::clMutex () [inline]

Constructor; creates and initializes the mutex.

5.30.2.2 clMutex::~~clMutex () [inline]

Destructor; destroys the mutex.

5.30.3 Member Function Documentation

5.30.3.1 `bool clMutex::Wait ()` [inline]

Lock mutex.

Process is blocked until mutex lock becomes available.

Returns:

Success

5.30.3.2 `bool clMutex::Release ()` [inline]

Unlock mutex.

Returns:

Success

5.30.3.3 `bool clMutex::TryLock ()` [inline]

Try locking mutex without blocking the calling process.

Returns:

Success

5.30.3.4 `pthread_mutex_t* clMutex::GetPtr ()` [inline]

Return pointer to mutex variable.

For use with condition variables.

Returns:

Pointer to pthread mutex variable

5.30.4 Member Data Documentation

5.30.4.1 `pthread_mutex_t clMutex::pthmMutex` [private]

5.31 clPthCond Class Reference

```
#include <PthCond.hh>
```

Public Member Functions

- [clPthCond](#) ()
- [~clPthCond](#) ()
- void [Wait](#) (pth_mutex_t *pthmCond)
- void [Wait](#) (pth_mutex_t *pthmCond, pth_event_t ptheCond)
- void [Notify](#) (int iNotifyAll)

Private Attributes

- pth_cond_t [pthcCond](#)

5.31.1 Constructor & Destructor Documentation

5.31.1.1 [clPthCond::clPthCond](#) () [inline]

5.31.1.2 [clPthCond::~~clPthCond](#) () [inline]

5.31.2 Member Function Documentation

5.31.2.1 void [clPthCond::Wait](#) (pth_mutex_t * *pthmCond*) [inline]

5.31.2.2 void [clPthCond::Wait](#) (pth_mutex_t * *pthmCond*, pth_event_t *ptheCond*) [inline]

5.31.2.3 void [clPthCond::Notify](#) (int *iNotifyAll*) [inline]

5.31.3 Member Data Documentation

5.31.3.1 pth_cond_t [clPthCond::pthcCond](#) [private]

5.32 clPthMutex Class Reference

```
#include <PthMutex.hh>
```

Public Member Functions

- [clPthMutex](#) ()
- [~clPthMutex](#) ()
- void [Wait](#) ()
- void [Release](#) ()
- pthread_t * [GetPtr](#) ()

Private Attributes

- pthread_t [pthmMutex](#)

5.32.1 Constructor & Destructor Documentation

5.32.1.1 [clPthMutex::clPthMutex](#) () [inline]

5.32.1.2 [clPthMutex::~~clPthMutex](#) () [inline]

5.32.2 Member Function Documentation

5.32.2.1 void [clPthMutex::Wait](#) () [inline]

5.32.2.2 void [clPthMutex::Release](#) () [inline]

5.32.2.3 pthread_t* [clPthMutex::GetPtr](#) () [inline]

5.32.3 Member Data Documentation

5.32.3.1 pthread_t [clPthMutex::pthmMutex](#) [private]

5.33 clReBuffer Class Reference

Class for splitting data into buffers of different sizes.

```
#include <ReBuffer.h>
```

Public Member Functions

- [clReBuffer](#) ()
- [clReBuffer](#) (const [clReBuffer](#) &CopySrc)
- [~clReBuffer](#) ()
- void [Put](#) (const float *, long)
Put data into FIFO.
- void [Put](#) (const double *, long)
- bool [Get](#) (float *, long)
Get data from FIFO.
- bool [Get](#) (double *, long)
- long [GetCount](#) () const
Get number of samples in FIFO.
- void [Clear](#) ()
Clear FIFO contents and reset buffer size.
- [clReBuffer](#) & [operator=](#) (const [clReBuffer](#) &)

Protected Member Functions

- void * [Index](#) (const std::type_info &, long)
- void * [GetPtr](#) (const std::type_info &)

Private Member Functions

- void [CheckSize](#) (long, long)

Private Attributes

- long [lSize](#)
- long [lPutIndex](#)
- long [lGetIndex](#)
- long [lCount](#)
- [clAlloc Buffer](#)

5.33.1 Detailed Description

Class for splitting data into buffers of different sizes.

It's implemented using dynamically growing circular FIFO buffer.

5.33.2 Constructor & Destructor Documentation

5.33.2.1 `clReBuffer::clReBuffer ()`

5.33.2.2 `clReBuffer::clReBuffer (const clReBuffer & CopySrc)` `[inline]`

5.33.2.3 `clReBuffer::~~clReBuffer ()`

5.33.3 Member Function Documentation

5.33.3.1 `void clReBuffer::CheckSize (long, long)` `[inline, private]`

5.33.3.2 `void * clReBuffer::Index (const std::type_info &, long)`
`[protected]`

5.33.3.3 `void * clReBuffer::GetPtr (const std::type_info &)` `[protected]`

5.33.3.4 `void clReBuffer::Put (const float *, long)`

Put data into FIFO.

Parameters:

fpSrcData Source buffer

lSrcCount Number of samples in source.

5.33.3.5 `void clReBuffer::Put (const double *, long)`

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.33.3.6 `bool clReBuffer::Get (float *, long)`

Get data from FIFO.

Returns false if there's not enough samples in FIFO to fill requested buffer.

Parameters:

fpDestData Destination buffer

lDestCount Number of samples to fetch

Returns:

Data available

5.33.3.7 bool clReBuffer::Get (double *, long)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.33.3.8 long clReBuffer::GetCount () const [inline]

Get number of samples in FIFO.

Returns:

Number of samples

5.33.3.9 void clReBuffer::Clear () [inline]

Clear FIFO contents and reset buffer size.

5.33.3.10 clReBuffer & clReBuffer::operator= (const clReBuffer &)**5.33.4 Member Data Documentation****5.33.4.1 long clReBuffer::lSize [private]****5.33.4.2 long clReBuffer::lPutIndex [private]****5.33.4.3 long clReBuffer::lGetIndex [private]****5.33.4.4 long clReBuffer::lCount [private]****5.33.4.5 clAlloc clReBuffer::Buffer [private]**

5.34 clReBufferT< TReBuffer_t > Class Template Reference

Template class for splitting data into buffers of different sizes.

```
#include <ReBufferT.hh>
```

Public Member Functions

- [clReBufferT](#) ()
- [clReBufferT](#) (const [clReBufferT](#) &CopySrc)
- [clReBufferT](#) (long lNewSize)
- [clReBufferT](#) (const TReBuffer_t *fpSrcData, long lSrcCount)
- [~clReBufferT](#) ()
- void [Put](#) (const TReBuffer_t *fpSrcData, long lSrcCount)
Put data into FIFO.
- void [Put](#) ([clReBufferT](#) &Src)
Put data into fifo from another FIFO.
- bool [Get](#) (TReBuffer_t *fpDestData, long lDestCount)
Get data from FIFO.
- long [GetCount](#) () const
Get number of samples in FIFO.
- long [Size](#) () const
- void [SetSize](#) (long lNewCount)
Set number of samples in FIFO.
- void [Resize](#) (long lNewCount)
Set number of samples in FIFO.
- void [Clear](#) ()
Clear FIFO contents and reset buffer size.
- [clReBufferT](#) & [operator=](#) (const [clReBufferT](#) &Src)
- TReBuffer_t & [operator\[\]](#) (long lIndex)

Protected Member Functions

- TReBuffer_t * [GetPtr](#) ()
- void [CopyGet](#) (TReBuffer_t *fpDestData, long lDestCount) const

Private Member Functions

- void [CheckSize](#) (long lDataCount)

Private Attributes

- long [lSize](#)
- long [lPutIndex](#)
- long [lGetIndex](#)
- long [lCount](#)
- [clDSPAlloc Buffer](#)

5.34.1 Detailed Description

```
template<class TReBuffer_t> class clReBufferT< TReBuffer_t >
```

Template class for splitting data into buffers of different sizes.

It's implemented using dynamically growing circular FIFO buffer. The ring buffer is linearized when linear access is requested.

5.34.2 Constructor & Destructor Documentation

5.34.2.1 `template<class TReBuffer_t> clReBufferT< TReBuffer_t
>::clReBufferT () [inline]`

5.34.2.2 `template<class TReBuffer_t> clReBufferT< TReBuffer_t
>::clReBufferT (const clReBufferT< TReBuffer_t > & CopySrc)
[inline]`

5.34.2.3 `template<class TReBuffer_t> clReBufferT< TReBuffer_t
>::clReBufferT (long lNewSize) [inline]`

5.34.2.4 `template<class TReBuffer_t> clReBufferT< TReBuffer_t
>::clReBufferT (const TReBuffer_t * fpSrcData, long lSrcCount)
[inline]`

5.34.2.5 `template<class TReBuffer_t> clReBufferT< TReBuffer_t
>::~~clReBufferT () [inline]`

5.34.3 Member Function Documentation

5.34.3.1 `template<class TReBuffer_t> void clReBufferT< TReBuffer_t
>::CheckSize (long lDataCount) [inline, private]`

5.34.3.2 `template<class TReBuffer_t> TReBuffer_t* clReBufferT<
TReBuffer_t >::GetPtr () [inline, protected]`

5.34.3.3 `template<class TReBuffer_t> void clReBufferT< TReBuffer_t
>::CopyGet (TReBuffer_t * fpDestData, long lDestCount) const
[inline, protected]`

5.34.3.4 `template<class TReBuffer_t> void clReBufferT< TReBuffer_t >::Put
(const TReBuffer_t * fpSrcData, long lSrcCount) [inline]`

Put data into FIFO.

Parameters:

fpSrcData Source buffer

lSrcCount Number of samples in source.

5.34.3.5 `template<class TReBuffer_t> void clReBufferT< TReBuffer_t >::Put
(clReBufferT< TReBuffer_t > & Src) [inline]`

Put data into fifo from another FIFO.

Parameters:

Src Source FIFO

5.34.3.6 `template<class TReBuffer_t> bool clReBufferT< TReBuffer_t
>::Get (TReBuffer_t * fpDestData, long lDestCount) [inline]`

Get data from FIFO.

Returns false if there's not enough samples in FIFO to fill requested buffer.

Parameters:

fpDestData Destination buffer

lDestCount Number of samples to fetch

Returns:

Data available

5.34.3.7 `template<class TReBuffer_t> long clReBufferT< TReBuffer_t
>::GetCount () const [inline]`

Get number of samples in FIFO.

Returns:

Number of samples

5.34.3.8 `template<class TReBuffer_t> long clReBufferT< TReBuffer_t
>::Size () const [inline]`

5.34.3.9 `template<class TReBuffer_t> void clReBufferT< TReBuffer_t
>::SetSize (long lNewCount) [inline]`

Set number of samples in FIFO.

Note:

This is destructive resize.

5.34.3.10 `template<class TReBuffer_t> void clReBufferT< TReBuffer_t
>::Resize (long lNewCount) [inline]`

Set number of samples in FIFO.

Note:

This is non-destructive resize.

5.34.3.11 `template<class TReBuffer_t> void clReBufferT< TReBuffer_t
>::Clear () [inline]`

Clear FIFO contents and reset buffer size.

5.34.3.12 `template<class TReBuffer_t> clReBufferT& clReBufferT< TReBuffer_t >::operator= (const clReBufferT< TReBuffer_t > & Src) [inline]`

5.34.3.13 `]`

`template<class TReBuffer_t> TReBuffer_t& clReBufferT< TReBuffer_t >::operator[] (long lIndex) [inline]`

5.34.4 Member Data Documentation

5.34.4.1 `template<class TReBuffer_t> long clReBufferT< TReBuffer_t >::lSize [private]`

5.34.4.2 `template<class TReBuffer_t> long clReBufferT< TReBuffer_t >::lPutIndex [private]`

5.34.4.3 `template<class TReBuffer_t> long clReBufferT< TReBuffer_t >::lGetIndex [private]`

5.34.4.4 `template<class TReBuffer_t> long clReBufferT< TReBuffer_t >::lCount [private]`

5.34.4.5 `template<class TReBuffer_t> clDSPAlloc clReBufferT< TReBuffer_t >::Buffer [private]`

5.35 clRecDecimator Class Reference

Recursive decimation filter class implementation.

```
#include <RecDecimator.hh>
```

Public Types

- enum `eFilterType` { `FILTER_TYPE_FFT` = 0, `FILTER_TYPE_FIR` = 1, `FILTER_TYPE_IIR` = 2 }

Public Member Functions

- `clRecDecimator` ()
- `~clRecDecimator` ()
- bool `Initialize` (long, long, const float *, float=0.0f, int=0)
Initialize decimator, decimation factor must be powers of two.
- bool `Initialize` (long, long, const double *, double=0.0, int=0)
- bool `Initialize` (long, long, const float *, float=0.0f, bool=false)
- bool `Initialize` (long, long, const double *, double=0.0, bool=false)
- void `Uninitialize` ()
Uninitialize decimator.
- void `Put` (const float *, long)
Feed data into decimator.
- void `Put` (const double *, long)
- bool `Get` (float *, long)
Get data from decimator.
- bool `Get` (double *, long)

Private Member Functions

- void `InitHalves` (double)

Private Attributes

- bool `bInitialized`
- int `iType`
Type of filter; 0 = FFT, 1 = FIR, 2 = IIR.
- long `lFactor`
Decimation factor.

- long [lFilterSize](#)
Size of input buffer.
- long [lSubRounds](#)
Number of recursive rounds.
- long [lDecSize](#)
Size of recursion buffer.
- bool [bpHalves](#) [RECDEC_MAX_SUB_ROUNDS]
- [clDSPAlloc DecBuf](#)
- [clFFTDecimator FFTDecBank](#) [RECDEC_MAX_SUB_ROUNDS]
- [clFIRDecimator FIRDecBank](#) [RECDEC_MAX_SUB_ROUNDS]
- [clIIRDecimator IIRDecBank](#) [RECDEC_MAX_SUB_ROUNDS]

5.35.1 Detailed Description

Recursive decimation filter class implementation.

Data is filtered using specified filter and decimated in recursive manner in rounds of factor 2.

5.35.2 Member Enumeration Documentation

5.35.2.1 enum [clRecDecimator::eFilterType](#)

Enumeration values:

FILTER_TYPE_FFT

FILTER_TYPE_FIR

FILTER_TYPE_IIR

5.35.3 Constructor & Destructor Documentation

5.35.3.1 [clRecDecimator::clRecDecimator](#) ()

5.35.3.2 [clRecDecimator::~~clRecDecimator](#) ()

5.35.4 Member Function Documentation

5.35.4.1 void [clRecDecimator::InitHalves](#) (double) [*private*]

5.35.4.2 bool [clRecDecimator::Initialize](#) (long, long, const float *, float = 0.0f, int = 0)

Initialize decimator, decimation factor must be powers of two.

You can re-initialize decimator without uninitializing first.

The NULL pointer is used select correct overloaded function matching input datatype.

Parameters:

DecFact Decimation factor

FiltSize Filter size (for FFT)

NullPtr NULL pointer

BandCenter Normalized center frequency of filter band

FilterType Filter type; 0 = FFT, 1 = FIR, 2 = IIR

5.35.4.3 bool clRecDecimator::Initialize (long, long, const double *, double = 0.0, int = 0)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.35.4.4 bool clRecDecimator::Initialize (long, long, const float *, float = 0.0f, bool = false)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.35.4.5 bool clRecDecimator::Initialize (long, long, const double *, double = 0.0, bool = false)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.35.4.6 void clRecDecimator::Uninitialize ()

Uninitialize decimator.

5.35.4.7 void clRecDecimator::Put (const float *, long)

Feed data into decimator.

Parameters:

SrcData Source data

SrcCount Source data count

5.35.4.8 void clRecDecimator::Put (const double *, long)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.35.4.9 bool clRecDecimator::Get (float *, long)

Get data from decimator.

Returns false if there's not enough data feeded into the decimator.

Parameters:

DestData Destination buffer

DestCount Number of samples

Returns:

Data was available?

5.35.4.10 bool clRecDecimator::Get (double *, long)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.35.5 Member Data Documentation

5.35.5.1 bool clRecDecimator::bInitialized [private]

5.35.5.2 int clRecDecimator::iType [private]

Type of filter; 0 = FFT, 1 = FIR, 2 = IIR.

5.35.5.3 long clRecDecimator::lFactor [private]

Decimation factor.

5.35.5.4 long clRecDecimator::lFilterSize [private]

Size of input buffer.

5.35.5.5 long clRecDecimator::lSubRounds [private]

Number of recursive rounds.

5.35.5.6 long clRecDecimator::lDecSize [private]

Size of recursion buffer.

- 5.35.5.7 **bool** [clRecDecimator::bpHalves](#)[RECDEC_MAX_SUB_ROUNDS]
[private]
- 5.35.5.8 [clDSPAlloc](#) [clRecDecimator::DecBuf](#) [private]
- 5.35.5.9 [clFFTDecimator](#) [clRecDecimator::FFTDecBank](#)[RECDEC_MAX_-
SUB_ROUNDS] [private]
- 5.35.5.10 [clFIRDecimator](#) [clRecDecimator::FIRDecBank](#)[RECDEC_MAX_-
SUB_ROUNDS] [private]
- 5.35.5.11 [clIIRDecimator](#) [clRecDecimator::IIRDecBank](#)[RECDEC_MAX_-
SUB_ROUNDS] [private]

5.36 clRecInterpolator Class Reference

Recursive interpolation filter class implementation.

```
#include <RecInterpolator.hh>
```

Public Types

- enum `eFilterType` { `FILTER_TYPE_FFT` = 0, `FILTER_TYPE_FIR` = 1, `FILTER_TYPE_IIR` = 2 }

Public Member Functions

- `clRecInterpolator` ()
- `~clRecInterpolator` ()
- bool `Initialize` (long, long, const float *, float=0.0f, int=0)
Initialize interpolator, interpolation factor must be powers of two.
- bool `Initialize` (long, long, const double *, double=0.0, int=0)
- bool `Initialize` (long, long, const float *, float=0.0f, bool=false)
- bool `Initialize` (long, long, const double *, double=0.0, bool=false)
- void `Uninitialize` ()
Uninitialize interpolator.
- void `Put` (const float *, long)
Feed data into interpolator.
- void `Put` (const double *, long)
- bool `Get` (float *, long)
Get data from interpolator.
- bool `Get` (double *, long)

Private Member Functions

- void `InitHalves` (double)

Private Attributes

- bool `bInitialized`
- int `iType`
Type filter; 0 = FFT, 1 = FIR, 2 = IIR.
- long `lFactor`
Interpolation factor.

- long [lFilterSize](#)
Size of input buffer.
- long [lSubRounds](#)
Number of recursive rounds.
- long [lIntSize](#)
Size of recursion buffer.
- bool [bpHalves](#) [RECINT_MAX_SUB_ROUNDS]
- [clDSPAlloc](#) [IntBuf](#)
- [clFFTInterpolator](#) [FFTIntBank](#) [RECINT_MAX_SUB_ROUNDS]
- [clFIRInterpolator](#) [FIRIntBank](#) [RECINT_MAX_SUB_ROUNDS]
- [clIIRInterpolator](#) [IIRIntBank](#) [RECINT_MAX_SUB_ROUNDS]

5.36.1 Detailed Description

Recursive interpolation filter class implementation.

Data is interpolated and filtered in recursive manner in rounds of factor 2.

5.36.2 Member Enumeration Documentation

5.36.2.1 enum [clRecInterpolator::eFilterType](#)

Enumeration values:

[FILTER_TYPE_FFT](#)

[FILTER_TYPE_FIR](#)

[FILTER_TYPE_IIR](#)

5.36.3 Constructor & Destructor Documentation

5.36.3.1 [clRecInterpolator::clRecInterpolator](#) ()

5.36.3.2 [clRecInterpolator::~~clRecInterpolator](#) ()

5.36.4 Member Function Documentation

5.36.4.1 void [clRecInterpolator::InitHalves](#) (double) [*private*]

5.36.4.2 bool [clRecInterpolator::Initialize](#) (long, long, const float *, float = 0.0f, int = 0)

Initialize interpolator, interpolation factor must be powers of two.

You can re-initialize interpolator without uninitializing first.

The NULL pointer is used select correct overloaded function matching input datatype.

Parameters:

IntFact Interpolation factor

FiltSize Filter size (for FFT)

NullPtr NULL pointer

BandCenter Normalized center frequency of filter band (destination)

FilterType Filter type; 0 = FFT, 1 = FIR, 2 = IIR

5.36.4.3 bool clRecInterpolator::Initialize (long, long, const double *, double = 0.0, int = 0)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.36.4.4 bool clRecInterpolator::Initialize (long, long, const float *, float = 0.0f, bool = false)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.36.4.5 bool clRecInterpolator::Initialize (long, long, const double *, double = 0.0, bool = false)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.36.4.6 void clRecInterpolator::Uninitialize ()

Uninitialize interpolator.

5.36.4.7 void clRecInterpolator::Put (const float *, long)

Feed data into interpolator.

Parameters:

SrcData Source data

SrcCount Source data count

5.36.4.8 void `clRecInterpolator::Put` (const double *, long)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.36.4.9 bool `clRecInterpolator::Get` (float *, long)

Get data from interpolator.

Returns false if there's not enough data feeded into the interpolator.

Parameters:

DestData Destination buffer

DestCount Number of samples

Returns:

Data was available?

5.36.4.10 bool `clRecInterpolator::Get` (double *, long)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.36.5 Member Data Documentation

5.36.5.1 bool `clRecInterpolator::bInitialized` [private]

5.36.5.2 int `clRecInterpolator::iType` [private]

Type filter; 0 = FFT, 1 = FIR, 2 = IIR.

5.36.5.3 long `clRecInterpolator::lFactor` [private]

Interpolation factor.

5.36.5.4 long `clRecInterpolator::lFilterSize` [private]

Size of input buffer.

5.36.5.5 long `clRecInterpolator::lSubRounds` [private]

Number of recursive rounds.

5.36.5.6 `long clRecInterpolator::IntSize` [private]

Size of recursion buffer.

5.36.5.7 `bool clRecInterpolator::bpHalves[RECINT_MAX_SUB_ROUNDS]`
[private]

5.36.5.8 `clDSPAlloc clRecInterpolator::IntBuf` [private]

5.36.5.9 `clFFTInterpolator clRecInterpolator::FFTIntBank[RECINT_MAX_SUB_ROUNDS]` [private]

5.36.5.10 `clFIRInterpolator clRecInterpolator::FIRIntBank[RECINT_MAX_SUB_ROUNDS]` [private]

5.36.5.11 `clIIRInterpolator clRecInterpolator::IIRIntBank[RECINT_MAX_SUB_ROUNDS]` [private]

5.37 clRWLock Class Reference

Class implementation of SUSv2 read-write locks.

```
#include <RWLock.hh>
```

Public Member Functions

- [clRWLock \(\)](#)
Constructor; creates and initializes the read-write lock.
- [~clRWLock \(\)](#)
Destructor; destroys the read-write lock.
- [bool WaitRead \(\)](#)
Obtain read-lock.
- [bool WaitWrite \(\)](#)
Obtain exclusive write-lock.
- [bool Release \(\)](#)
Release lock.
- [bool TryRead \(\)](#)
Try to obtain read-lock (non-blocking).
- [bool TryWrite \(\)](#)
Try to obtain exclusive write-lock (non-blocking).

Private Attributes

- `pthread_rwlock_t` [ptrwlLock](#)

5.37.1 Detailed Description

Class implementation of SUSv2 read-write locks.

5.37.2 Constructor & Destructor Documentation

5.37.2.1 clRWLock::clRWLock () [inline]

Constructor; creates and initializes the read-write lock.

5.37.2.2 clRWLock::~~clRWLock () [inline]

Destructor; destroys the read-write lock.

5.37.3 Member Function Documentation

5.37.3.1 bool clRWLock::WaitRead () [inline]

Obtain read-lock.

There can be multiple read-locks at the same time.

Returns:

Success

5.37.3.2 bool clRWLock::WaitWrite () [inline]

Obtain exclusive write-lock.

There can be only one write-lock and no read-locks at the same time.

Returns:

Success

5.37.3.3 bool clRWLock::Release () [inline]

Release lock.

Returns:

Success

5.37.3.4 bool clRWLock::TryRead () [inline]

Try to obtain read-lock (non-blocking).

Returns:

Success

5.37.3.5 bool clRWLock::TryWrite () [inline]

Try to obtain exclusive write-lock (non-blocking).

Returns:

Success

5.37.4 Member Data Documentation

5.37.4.1 pthread_rwlock_t [clRWLock::ptrwlLock](#) [private]

5.38 clSemaphore Class Reference

Class implementation of POSIX counting semaphores.

```
#include <Semaphore.hh>
```

Public Member Functions

- [clSemaphore](#) ()
Constructor; create and initialize the semaphore.
- [clSemaphore](#) (unsigned int uiSemValue)
Constructor; create and initialize the semaphore.
- [~clSemaphore](#) ()
Destructor; destroy the semaphore.
- bool [Initialize](#) (unsigned int uiSemValue)
Initialize the semaphore to specified value.
- void [Wait](#) ()
Wait until semaphore count becomes non-zero and then decrement the count.
- bool [TryWait](#) ()
Test if semaphore count is non-zero, if it is, then count is decremented.
- bool [Post](#) ()
Post (increment) semaphore.
- int [GetValue](#) ()
Get value of semaphore.

Private Attributes

- sem_t [semSemaphore](#)

5.38.1 Detailed Description

Class implementation of POSIX counting semaphores.

5.38.2 Constructor & Destructor Documentation

5.38.2.1 `clSemaphore::clSemaphore ()` [inline]

Constructor; create and initialize the semaphore.

Semaphore value is initialized to zero.

5.38.2.2 `clSemaphore::clSemaphore (unsigned int uiSemValue)` [inline]

Constructor; create and initialize the semaphore.

Semaphore is initialized to caller specified value.

Parameters:

uiSemValue Initial semaphore value

5.38.2.3 `clSemaphore::~clSemaphore ()` [inline]

Destructor; destroy the semaphore.

5.38.3 Member Function Documentation

5.38.3.1 `bool clSemaphore::Initialize (unsigned int uiSemValue)` [inline]

Initialize the semaphore to specified value.

This can be used to explicitly set the semaphore value.

Parameters:

uiSemValue New value of semaphore

Returns:

Success

5.38.3.2 `void clSemaphore::Wait ()` [inline]

Wait until semaphore count becomes non-zero and then decrement the count.

5.38.3.3 `bool clSemaphore::TryWait ()` [inline]

Test if semaphore count is non-zero, if it is, then count is decremented.

5.38.3.4 bool clSemaphore::Post () [inline]

Post (increment) semaphore.

Returns:

Success

5.38.3.5 int clSemaphore::GetValue () [inline]

Get value of semaphore.

Returns:

Semaphore count

5.38.4 Member Data Documentation

5.38.4.1 sem_t clSemaphore::semSemaphore [private]

5.39 clTransform4 Class Reference

Decimation-in-frequency radix-2/4 transform.

```
#include <Transform4.hh>
```

Public Member Functions

- void [cdft](#) (long, long, float *, long *, float *)
Complex DFT.
- void [cdft](#) (long, long, double *, long *, double *)
- void [rdft](#) (long, long, float *, long *, float *)
Real DFT.
- void [rdft](#) (long, long, double *, long *, double *)
- void [ddct](#) (long, long, float *, long *, float *)
DCT.
- void [ddct](#) (long, long, double *, long *, double *)
- void [ddst](#) (long, long, float *, long *, float *)
DST.
- void [ddst](#) (long, long, double *, long *, double *)
- void [dfct](#) (long, float *, float *, long *, float *)
CT of RDFT.
- void [dfct](#) (long, double *, double *, long *, double *)
- void [dfst](#) (long, float *, float *, long *, float *)
ST of RDFT.
- void [dfst](#) (long, double *, double *, long *, double *)

Private Member Functions

- void [makewt](#) (long, long *, float *)
- void [makewt](#) (long, long *, double *)
- void [makect](#) (long, long *, float *)
- void [makect](#) (long, long *, double *)
- void [bitrv2](#) (long, long *, float *)
- void [bitrv2](#) (long, long *, double *)
- void [bitrv2conj](#) (long, long *, float *)
- void [bitrv2conj](#) (long, long *, double *)
- void [cftfsub](#) (long, float *, float *)
- void [cftfsub](#) (long, double *, double *)
- void [cftbsub](#) (long, float *, float *)

- void [cftbsub](#) (long, double *, double *)
- void [cft1st](#) (long, float *, float *)
- void [cft1st](#) (long, double *, double *)
- void [cftmdl](#) (long, long, float *, float *)
- void [cftmdl](#) (long, long, double *, double *)
- void [rftbsub](#) (long, float *, long, float *)
- void [rftbsub](#) (long, double *, long, double *)
- void [rftbsub](#) (long, float *, long, float *)
- void [rftbsub](#) (long, double *, long, double *)
- void [dctsub](#) (long, float *, long, float *)
- void [dctsub](#) (long, double *, long, double *)
- void [dstsub](#) (long, float *, long, float *)
- void [dstsub](#) (long, double *, long, double *)

5.39.1 Detailed Description

Decimation-in-frequency radix-2/4 transform.

Author:

Takuya OOURA
Jussi Laako

Note:

DFT parts of this class has been wrapped into [clDSPOp](#) with easier interface.

5.39.2 Member Function Documentation

- 5.39.2.1 **void clTransform4::makewt (long, long *, float *)** [private]
- 5.39.2.2 **void clTransform4::makewt (long, long *, double *)** [private]
- 5.39.2.3 **void clTransform4::makeect (long, long *, float *)** [private]
- 5.39.2.4 **void clTransform4::makeect (long, long *, double *)** [private]
- 5.39.2.5 **T4_INLINE void clTransform4::bitrv2 (long, long *, float *)**
[private]
- 5.39.2.6 **T4_INLINE void clTransform4::bitrv2 (long, long *, double *)**
[private]
- 5.39.2.7 **T4_INLINE void clTransform4::bitrv2conj (long, long *, float *)**
[private]
- 5.39.2.8 **T4_INLINE void clTransform4::bitrv2conj (long, long *, double *)**
[private]
- 5.39.2.9 **T4_INLINE void clTransform4::cftsub (long, float *, float *)**
[private]
- 5.39.2.10 **T4_INLINE void clTransform4::cftsub (long, double *, double *)**
[private]
- 5.39.2.11 **T4_INLINE void clTransform4::cftbsub (long, float *, float *)**
[private]
- 5.39.2.12 **T4_INLINE void clTransform4::cftbsub (long, double *, double *)**
[private]
- 5.39.2.13 **T4_INLINE void clTransform4::cft1st (long, float *, float *)**
[private]
- 5.39.2.14 **T4_INLINE void clTransform4::cft1st (long, double *, double *)**
[private]
- 5.39.2.15 **T4_INLINE void clTransform4::cftmdl (long, long, float *, float *)**
[private]
- 5.39.2.16 **T4_INLINE void clTransform4::cftmdl (long, long, double *, double *)**
[private]
- 5.39.2.17 **T4_INLINE void clTransform4::rftsub (long, float *, long, float *)**
[private]
- 5.39.2.18 **T4_INLINE void clTransform4::rftsub (long, double *, long, double *)**
[private]
- 5.39.2.19 **T4_INLINE void clTransform4::rftbsub (long, float *, long, float *)**
[private]
- 5.39.2.20 **T4_INLINE void clTransform4::rftbsub (long, double *, long, double *)**
[private]
- 5.39.2.21 **T4_INLINE void clTransform4::dctsub (long, float *, long, float *)**

Forward:

$$X(k) = \sum_{j=0}^{N-1} x(j) \exp(2\pi i j k / N), 0 \leq k \leq N-1$$

Inverse:

$$X(k) = \sum_{j=0}^{N-1} x(j) \exp(-2\pi i j k / N), 0 \leq k \leq N-1$$

Input data:

$$\begin{cases} a(2j) = \Re_x(j) \\ a(2j+1) = \Im_x(j) \end{cases}, 0 \leq j \leq N-1$$

Output data:

$$\begin{cases} a(2k) = \Re_X(k) \\ a(2k+1) = \Im_X(k) \end{cases}, 0 \leq k \leq N-1$$

Work area size:

$$2 + \sqrt{N}$$

Scaling factor:

$$\frac{1}{N}$$

Parameters:

Size Window size, 2*N

Direction Direction of transform; 1 = fwd, -1 = inv

Data Source & destination data, size 2*N

WorkArea Working area, [0] = 0 to initialize

CosSin cos/sin table, size N/2

5.39.2.26 void clTransform4::cdft (long, long, double *, long *, double *)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.39.2.27 void clTransform4::rdft (long, long, float *, long *, float *)

Real DFT.

Forward:

$$\begin{cases} \Re_X(k) = \sum_{j=0}^{N-1} a(j) \cos(2\pi j k / N) \\ \Im_X(k) = \sum_{j=0}^{N-1} a(j) \sin(2\pi j k / N) \end{cases}, 0 \leq k < N/2$$

Inverse:

$$a(k) = \frac{(\Re_X(0) + \Re_X(\frac{N}{2}) \cos(\pi k))}{2} + \sum_{j=1}^{N/2-1} \Re_X(j) \cos(2\pi j k / N) + \sum_{j=1}^{N/2-1} \Im_X(j) \sin(2\pi j k / N), 0 \leq k \leq N-1$$

Output data:

$$\left| \begin{array}{ll} a(2k) = \Re_X(k) & , 0 \leq k < N/2 \\ a(2k+1) = \Im_X(k) & , 0 < k < N/2 \\ a(1) = \Re_X(N/2) \end{array} \right.$$

Input data:

$$\left| \begin{array}{ll} a(2j) = \Re_X(j) & , 0 \leq j < N/2 \\ a(2j+1) = \Im_X(j) & , 0 < j < N/2 \\ a(1) = \Re_X(N/2) \end{array} \right.$$

Work area size:

$$2 + \sqrt{\frac{N}{2}}$$

Scaling factor:

$$\frac{2}{N}$$

Parameters:

Size Window size, N

Direction Direction of transform; 1 = fwd, -1 = inv

Data Source & destination data, size N

WorkArea Working area, [0] = 0 to initialize

CosSin cos/sin table, size N/2

5.39.2.28 void clTransform4::rdft (long, long, double *, long *, double *)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.39.2.29 void clTransform4::ddct (long, long, float *, long *, float *)

DCT.

Forward:

$$C(k) = \sum_{j=0}^{N-1} a(j) \cos(\pi j(k+1/2)/N), 0 \leq k \leq N-1$$

Inverse:

$$C(k) = \sum_{j=0}^{N-1} a(j) \cos(\pi(j+1/2)k/N), 0 \leq k \leq N-1$$

Input/output data:

$$a(k) = C(k), 0 \leq k \leq N-1$$

Work area size:

$$2 + \sqrt{\frac{N}{2}}$$

Scaling factors:

$$(0)0.5, \frac{2}{N}$$

Parameters:

Size Window size, N

Direction Direction of transform; 1 = fwd, -1 = inv

Data Source & destination data, size N

WorkArea Working area, [0] = 0 to initialize

CosSin cos/sin table, size N*5/4

5.39.2.30 void clTransform4::ddct (long, long, double *, long *, double *)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.39.2.31 void clTransform4::ddst (long, long, float *, long *, float *)

DST.

Forward:

$$S(k) = \sum_{j=1}^N A(j) \sin(\pi j(k + 1/2)/N), 0 \leq k \leq N - 1$$

Inverse:

$$S(k) = \sum_{j=0}^{N-1} a(j) \sin(\pi(j + 1/2)k/N), 0 < k \leq N$$

Input data (forward):

$$\left| \begin{array}{l} a(j) = A(j), 0 < j < N \\ a(0) = A(N) \end{array} \right.$$

Output data (forward):

$$a(k) = S(k), 0 \leq k \leq N - 1$$

Output data (inverse):

$$\left| \begin{array}{l} a(k) = S(k), 0 < k < N \\ a(0) = S(N) \end{array} \right.$$

Work area size:

$$2 + \sqrt{\frac{N}{2}}$$

Scaling factors:

$$(0)0.5, \frac{2}{N}$$

Parameters:

Size Window size, N

Direction Direction of transform; 1 = fwd, -1 = inv

Data Source & destination data, size N

WorkArea Working area, [0] = 0 to initialize

CosSin cos/sin table, size N*5/4

5.39.2.32 void clTransform4::ddst (long, long, double *, long *, double *)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.39.2.33 void clTransform4::dfct (long, float *, float *, long *, float *)

CT of RDFT.

Definition:

$$C(k) = \sum_{j=0}^N a(j) \cos(\pi j k / N), 0 \leq k \leq N$$

Output data:

$$a(k) = C(k), 0 \leq k \leq N$$

Work area size:

$$2 + \sqrt{\frac{N}{4}}$$

Scaling factors:

$$(0)0.5, (N)0.5, \frac{2}{N}$$

Parameters:

Size Window size, N

Data Source & destination data, size N+1

Scratch Scratch pad, size N/2+1

WorkArea Working area, [0] = 0 to initialize

CosSin cos/sin table, size N*5/8

5.39.2.34 void clTransform4::dfct (long, double *, double *, long *, double *)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.39.2.35 void clTransform4::dfst (long, float *, float *, long *, float *)

ST of RDFT.

Definition:

$$S(k) = \sum_{j=1}^{N-1} a(j) \sin(\pi j k / N), 0 < k < N$$

Output data:

$$a(k) = S(k), 0 < k < N$$

Work area size:

$$2 + \sqrt{\frac{N}{4}}$$

Scaling factor:

$$\frac{2}{N}$$

Parameters:

Size Window size, N

Data Source & destination data, size N+1

Scratch Scratch pad, size N/2

WorkArea Working area, [0] = 0 to initialize

CosSin cos/sin table, size N*5/8

5.39.2.36 void clTransform4::dfst (long, double *, double *, long *, double *)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

5.40 clTransform8 Class Reference

Decimation-in-frequency radix-2/4/8 transform.

```
#include <Transform8.hh>
```

Public Member Functions

- void [cdft](#) (long, long, float *, long *, float *)
- void [cdft](#) (long, long, double *, long *, double *)
- void [rdft](#) (long, long, float *, long *, float *)
- void [rdft](#) (long, long, double *, long *, double *)
- void [ddct](#) (long, long, float *, long *, float *)
- void [ddct](#) (long, long, double *, long *, double *)
- void [ddst](#) (long, long, float *, long *, float *)
- void [ddst](#) (long, long, double *, long *, double *)
- void [dfct](#) (long, float *, float *, long *, float *)
- void [dfct](#) (long, double *, double *, long *, double *)
- void [dfst](#) (long, float *, float *, long *, float *)
- void [dfst](#) (long, double *, double *, long *, double *)

Private Member Functions

- void [makewt](#) (long, long *, float *)
- void [makewt](#) (long, long *, double *)
- void [makect](#) (long, long *, float *)
- void [makect](#) (long, long *, double *)
- void [bitrv2](#) (long, long *, float *)
- void [bitrv2](#) (long, long *, double *)
- void [bitrv2conj](#) (long, long *, float *)
- void [bitrv2conj](#) (long, long *, double *)
- void [cftfsub](#) (long, float *, float *)
- void [cftfsub](#) (long, double *, double *)
- void [cftbsub](#) (long, float *, float *)
- void [cftbsub](#) (long, double *, double *)
- void [cft1st](#) (long, float *, float *)
- void [cft1st](#) (long, double *, double *)
- void [cftmdl](#) (long, long, float *, float *)
- void [cftmdl](#) (long, long, double *, double *)
- void [rftfsub](#) (long, float *, long, float *)
- void [rftfsub](#) (long, double *, long, double *)
- void [rftbsub](#) (long, float *, long, float *)
- void [rftbsub](#) (long, double *, long, double *)
- void [dctsub](#) (long, float *, long, float *)
- void [dctsub](#) (long, double *, long, double *)
- void [dstsub](#) (long, float *, long, float *)
- void [dstsub](#) (long, double *, long, double *)

5.40.1 Detailed Description

Decimation-in-frequency radix-2/4/8 transform.

Author:

Takuya OOURA
Jussi Laako

Note:

See [clTransform4](#) for details.

5.40.2 Member Function Documentation

5.40.2.1 void clTransform8::makewt (long, long *, float *) [private]

5.40.2.2 void clTransform8::makewt (long, long *, double *) [private]

5.40.2.3 void clTransform8::makeect (long, long *, float *) [private]

5.40.2.4 void clTransform8::makeect (long, long *, double *) [private]

5.40.2.5 T8_INLINE void clTransform8::bitrv2 (long, long *, float *)
[private]

5.40.2.6 T8_INLINE void clTransform8::bitrv2 (long, long *, double *)
[private]

5.40.2.7 T8_INLINE void clTransform8::bitrv2conj (long, long *, float *)
[private]

5.40.2.8 T8_INLINE void clTransform8::bitrv2conj (long, long *, double *)
[private]

5.40.2.9 T8_INLINE void clTransform8::cftfsub (long, float *, float *)
[private]

5.40.2.10 T8_INLINE void clTransform8::cftfsub (long, double *, double *)
[private]

5.40.2.11 T8_INLINE void clTransform8::cftbsub (long, float *, float *)
[private]

5.40.2.12 T8_INLINE void clTransform8::cftbsub (long, double *, double *)
[private]

5.40.2.13 T8_INLINE void clTransform8::cft1st (long, float *, float *)
[private]

5.40.2.14 T8_INLINE void clTransform8::cft1st (long, double *, double *)
[private]

5.40.2.15 T8_INLINE void clTransform8::cftmdl (long, long, float *, float *)
[private]

5.40.2.16 T8_INLINE void clTransform8::cftmdl (long, long, double *, double
*) [private]

5.40.2.17 T8_INLINE void clTransform8::rftfsub (long, float *, long, float *)
[private]

5.40.2.18 T8_INLINE void clTransform8::rftfsub (long, double *, long, double
*) [private]

Generated on Tue Mar 2 19:56:37 2004 for libDSP by Doxygen

5.40.2.19 T8_INLINE void clTransform8::rftbsub (long, float *, long, float *)
[private]

5.40.2.20 T8_INLINE void clTransform8::rftbsub (long, double *, long, double
*) [private]

5.40.2.21 T8_INLINE void clTransform8::dctsub (long, float *, long, float *)

5.41 clTransformS Class Reference

Decimation-in-frequency split-radix transform.

```
#include <TransformS.hh>
```

Public Member Functions

- [clTransformS](#) ()
- [~clTransformS](#) ()
- void [cdft](#) (long, long, float *, long *, float *)
- void [cdft](#) (long, long, double *, long *, double *)
- void [rdft](#) (long, long, float *, long *, float *)
- void [rdft](#) (long, long, double *, long *, double *)
- void [ddct](#) (long, long, float *, long *, float *)
- void [ddct](#) (long, long, double *, long *, double *)
- void [ddst](#) (long, long, float *, long *, float *)
- void [ddst](#) (long, long, double *, long *, double *)
- void [dfct](#) (long, float *, float *, long *, float *)
- void [dfct](#) (long, double *, double *, long *, double *)
- void [dfst](#) (long, float *, float *, long *, float *)
- void [dfst](#) (long, double *, double *, long *, double *)

Private Member Functions

- void [makeipt](#) (long, long *)
- void [makewt](#) (long, long *, float *)
- void [makewt](#) (long, long *, double *)
- void [makect](#) (long, long *, float *)
- void [makect](#) (long, long *, double *)
- void [cftfsub](#) (long, float *, long *, long, float *)
- void [cftfsub](#) (long, double *, long *, long, double *)
- void [cftbsub](#) (long, float *, long *, long, float *)
- void [cftbsub](#) (long, double *, long *, long, double *)
- void [bitrv2](#) (long, long *, float *)
- void [bitrv2](#) (long, long *, double *)
- void [bitrv2conj](#) (long, long *, float *)
- void [bitrv2conj](#) (long, long *, double *)
- void [bitrv216](#) (float *)
- void [bitrv216](#) (double *)
- void [bitrv216neg](#) (float *)
- void [bitrv216neg](#) (double *)
- void [bitrv208](#) (float *)
- void [bitrv208](#) (double *)
- void [bitrv208neg](#) (float *)
- void [bitrv208neg](#) (double *)

- void [cftf1st](#) (long, float *, float *)
- void [cftf1st](#) (long, double *, double *)
- void [cftb1st](#) (long, float *, float *)
- void [cftb1st](#) (long, double *, double *)
- void [cftrec4](#) (long, float *, long, float *)
- void [cftrec4](#) (long, double *, long, double *)
- long [cfttree](#) (long, long, long, float *, long, float *)
- long [cfttree](#) (long, long, long, double *, long, double *)
- void [cftleaf](#) (long, long, float *, long, float *)
- void [cftleaf](#) (long, long, double *, long, double *)
- void [cftmdl1](#) (long, float *, float *)
- void [cftmdl1](#) (long, double *, double *)
- void [cftmdl2](#) (long, float *, float *)
- void [cftmdl2](#) (long, double *, double *)
- void [cftfx41](#) (long, float *, long, float *)
- void [cftfx41](#) (long, double *, long, double *)
- void [cftf161](#) (float *, float *)
- void [cftf161](#) (double *, double *)
- void [cftf162](#) (float *, float *)
- void [cftf162](#) (double *, double *)
- void [cftf081](#) (float *, float *)
- void [cftf081](#) (double *, double *)
- void [cftf082](#) (float *, float *)
- void [cftf082](#) (double *, double *)
- void [cftf040](#) (float *)
- void [cftf040](#) (double *)
- void [cftb040](#) (float *)
- void [cftb040](#) (double *)
- void [cftx020](#) (float *)
- void [cftx020](#) (double *)
- void [rftfsub](#) (long, float *, long, float *)
- void [rftfsub](#) (long, double *, long, double *)
- void [rftbsub](#) (long, float *, long, float *)
- void [rftbsub](#) (long, double *, long, double *)
- void [dctsub](#) (long, float *, long, float *)
- void [dctsub](#) (long, double *, long, double *)
- void [dstsub](#) (long, float *, long, float *)
- void [dstsub](#) (long, double *, long, double *)

5.41.1 Detailed Description

Decimation-in-frequency split-radix transform.

Author:

Takuya OOURA
Jussi Laako

Note:

See [clTransform4](#) for details.

5.41.2 Constructor & Destructor Documentation

5.41.2.1 `clTransformS::clTransformS ()` [inline]

5.41.2.2 `clTransformS::~~clTransformS ()` [inline]

5.41.3 Member Function Documentation

5.41.3.1 `void clTransformS::makeipt (long, long *)` [private]

5.41.3.2 `void clTransformS::makewt (long, long *, float *)` [private]

5.41.3.3 `void clTransformS::makewt (long, long *, double *)` [private]

5.41.3.4 `void clTransformS::makeect (long, long *, float *)` [private]

5.41.3.5 `void clTransformS::makeect (long, long *, double *)` [private]

5.41.3.6 `void clTransformS::cftftsub (long, float *, long *, long, float *)`
[private]

5.41.3.7 `void clTransformS::cftftsub (long, double *, long *, long, double *)`
[private]

5.41.3.8 `void clTransformS::cftbsub (long, float *, long *, long, float *)`
[private]

5.41.3.9 `void clTransformS::cftbsub (long, double *, long *, long, double *)`
[private]

5.41.3.10 `TS_INLINE void clTransformS::bitrv2 (long, long *, float *)`
[private]

5.41.3.11 `TS_INLINE void clTransformS::bitrv2 (long, long *, double *)`
[private]

5.41.3.12 `TS_INLINE void clTransformS::bitrv2conj (long, long *, float *)`
[private]

5.41.3.13 `TS_INLINE void clTransformS::bitrv2conj (long, long *, double *)`
[private]

5.41.3.14 `TS_INLINE void clTransformS::bitrv216 (float *)` [private]

5.41.3.15 `TS_INLINE void clTransformS::bitrv216 (double *)` [private]

5.41.3.16 `TS_INLINE void clTransformS::bitrv216neg (float *)` [private]

5.41.3.17 `TS_INLINE void clTransformS::bitrv216neg (double *)`
[private]

5.41.3.18 `TS_INLINE void clTransformS::bitrv208 (float *)` [private]

5.41.3.19 `TS_INLINE void clTransformS::bitrv208 (double *)` [private]

5.41.3.20 `TS_INLINE void clTransformS::bitrv208neg (float *)` [private]

5.41.3.21 `TS_INLINE void clTransformS::bitrv208neg (double *)`
[private]

5.42 clUserThreads Class Reference

Public Member Functions

- [clUserThreads \(\)](#)
- void * [Thread1](#) (void *)
- void * [Thread2](#) (void *)
- void * [Thread3](#) (void *)
- void [Stop](#) ()

Private Attributes

- volatile bool [bRun](#)

5.42.1 Constructor & Destructor Documentation

5.42.1.1 `clUserThreads::clUserThreads () [inline]`

5.42.2 Member Function Documentation

5.42.2.1 `void * clUserThreads::Thread1 (void *)`

5.42.2.2 `void * clUserThreads::Thread2 (void *)`

5.42.2.3 `void * clUserThreads::Thread3 (void *)`

5.42.2.4 `void clUserThreads::Stop () [inline]`

5.42.3 Member Data Documentation

5.42.3.1 `volatile bool clUserThreads::bRun [private]`

Chapter 6

libDSP File Documentation

6.1 Alloc.hh File Reference

```
#include <stdlib.h>
#include <sys/types.h>
#include <sys/mman.h>
#include <string>
#include <stdexcept>
```

Classes

- class [clAlloc](#)
Class for memory allocation operations.

Defines

- #define [ALLOC_USE_ALIGN](#)
- #define [ALLOC_ALIGNMENT](#) 32

6.1.1 Define Documentation

6.1.1.1 #define ALLOC_USE_ALIGN

6.1.1.2 #define ALLOC_ALIGNMENT 32

6.2 cdspop.cc File Reference

```
#include "dsp/dspop.h"
#include "dsp/DSPOp.hh"
#include "dsp/IIRCascade.hh"
#include "dsp/FFTDecimator.hh"
#include "dsp/FIRDecimator.hh"
#include "dsp/IIRDecimator.hh"
#include "dsp/RecDecimator.hh"
#include "dsp/FFTInterpolator.hh"
#include "dsp/FIRInterpolator.hh"
#include "dsp/IIRInterpolator.hh"
#include "dsp/RecInterpolator.hh"
#include "dsp/Filter.hh"
#include "dsp/ReBuffer.hh"
```

Functions

- void [dsp_init](#) ()
Initialize DSP library.
- signed long [dsp_roundf](#) (float fSrc)
cdSPOp::Round
- signed long [dsp_round](#) (double dSrc)
- void [dsp_addf](#) (float *fpDest, float fSrc, long lCount)
cdSPOp::Add
- void [dsp_add](#) (double *dpDest, double dSrc, long lCount)
- void [dsp_caddf](#) (stpSCplx spDest, stSCplx sSrc, long lCount)
- void [dsp_cadd](#) (stpDCplx spDest, stDCplx sSrc, long lCount)
- void [dsp_add2f](#) (float *fpDest, const float *fpSrc, long lCount)
- void [dsp_add2](#) (double *dpDest, const double *dpSrc, long lCount)
- void [dsp_cadd2f](#) (stpSCplx spDest, const stpSCplx spSrc, long lCount)
- void [dsp_cadd2](#) (stpDCplx spDest, const stpDCplx spSrc, long lCount)
- void [dsp_add3f](#) (float *fpDest, const float *fpSrc1, const float *fpSrc2, long lCount)
- void [dsp_add3](#) (double *dpDest, const double *dpSrc1, const double *dpSrc2, long lCount)
- void [dsp_cadd3f](#) (stpSCplx spDest, const stpSCplx spSrc1, const stpSCplx spSrc2, long lCount)

- void `dsp_cadd3` (`stpDCplx` spDest, const `stpDCplx` spSrc1, const `stpDCplx` spSrc2, long lCount)
- void `dsp_subf` (float *fpDest, float fSrc, long lCount)
cdSPOp::Sub
- void `dsp_sub` (double *dpDest, double dSrc, long lCount)
- void `dsp_csubf` (`stpSCplx` spDest, `stSCplx` sSrc, long lCount)
- void `dsp_csub` (`stpDCplx` spDest, `stDCplx` sSrc, long lCount)
- void `dsp_sub2f` (float *fpDest, const float *fpSrc, long lCount)
- void `dsp_sub2` (double *dpDest, const double *dpSrc, long lCount)
- void `dsp_csub2f` (`stpSCplx` spDest, const `stpSCplx` spSrc, long lCount)
- void `dsp_csub2` (`stpDCplx` spDest, const `stpDCplx` spSrc, long lCount)
- void `dsp_sub3f` (float *fpDest, const float *fpSrc1, const float *fpSrc2, long lCount)
- void `dsp_sub3` (double *dpDest, const double *dpSrc1, const double *dpSrc2, long lCount)
- void `dsp_csub3f` (`stpSCplx` spDest, const `stpSCplx` spSrc1, const `stpSCplx` spSrc2, long lCount)
- void `dsp_csub3` (`stpDCplx` spDest, const `stpDCplx` spSrc1, const `stpDCplx` spSrc2, long lCount)
- void `dsp_mulf` (float *fpDest, float fSrc, long lCount)
cdSPOp::Mul
- void `dsp_mul` (double *dpDest, double dSrc, long lCount)
- void `dsp_chmulf` (`stpSCplx` spDest, float fSrc, long lCount)
- void `dsp_chmul` (`stpDCplx` spDest, double dSrc, long lCount)
- void `dsp_emulf` (`stpSCplx` spDest, `stSCplx` sSrc, long lCount)
- void `dsp_emul` (`stpDCplx` spDest, `stDCplx` sSrc, long lCount)
- void `dsp_mulf_nip` (float *fpDest, const float *fpSrc1, float fSrc2, long lCount)
- void `dsp_mul_nip` (double *dpDest, const double *dpSrc1, double dSrc2, long lCount)
- void `dsp_mul2f` (float *fpDest, const float *fpSrc, long lCount)
- void `dsp_mul2` (double *dpDest, const double *dpSrc, long lCount)
- void `dsp_chmul2f` (`stpSCplx` spDest, const float *fpSrc, long lCount)
- void `dsp_chmul2` (`stpDCplx` spDest, const double *dpSrc, long lCount)
- void `dsp_emul2f` (`stpSCplx` spDest, const `stpSCplx` spSrc, long lCount)
- void `dsp_emul2` (`stpDCplx` spDest, const `stpDCplx` spSrc, long lCount)
- void `dsp_mul3f` (float *fpDest, const float *fpSrc1, const float *fpSrc2, long lCount)
- void `dsp_mul3` (double *dpDest, const double *dpSrc1, const double *dpSrc2, long lCount)
- void `dsp_emul3f` (`stpSCplx` spDest, const `stpSCplx` spSrc1, const `stpSCplx` spSrc2, long lCount)
- void `dsp_emul3` (`stpDCplx` spDest, const `stpDCplx` spSrc1, const `stpDCplx` spSrc2, long lCount)
- void `dsp_ccmulf` (`stpSCplx` spDest, const `stpSCplx` spSrc, long lCount)
- void `dsp_ccmul` (`stpDCplx` spDest, const `stpDCplx` spSrc, long lCount)

- void `dsp_ccmulf_nip` (`stpSCplx` spDest, const `stpSCplx` spSrc1, const `stpSCplx` spSrc2, long lCount)
- void `dsp_ccmul_nip` (`stpDCplx` spDest, const `stpDCplx` spSrc1, const `stpDCplx` spSrc2, long lCount)
- void `dsp_divf` (float *fpDest, float fSrc, long lCount)

clDSPOp::Div

- void `dsp_div` (double *dpDest, double dSrc, long lCount)
- void `dsp_cdivf` (`stpSCplx` spDest, `stSCplx` sSrc, long lCount)
- void `dsp_cdiv` (`stpDCplx` spDest, `stDCplx` sSrc, long lCount)
- void `dsp_div2f` (float *fpDest, const float *fpSrc, long lCount)
- void `dsp_div2` (double *dpDest, const double *dpSrc, long lCount)
- void `dsp_cdiv2f` (`stpSCplx` spDest, const `stpSCplx` spSrc, long lCount)
- void `dsp_cdiv2` (`stpDCplx` spDest, const `stpDCplx` spSrc, long lCount)
- void `dsp_div3f` (float *fpDest, const float *fpSrc1, const float *fpSrc2, long lCount)
- void `dsp_div3` (double *dpDest, const double *dpSrc1, const double *dpSrc2, long lCount)
- void `dsp_cdiv3f` (`stpSCplx` spDest, const `stpSCplx` spSrc1, const `stpSCplx` spSrc2, long lCount)
- void `dsp_cdiv3` (`stpDCplx` spDest, const `stpDCplx` spSrc1, const `stpDCplx` spSrc2, long lCount)
- void `dsp_div1xf` (float *fpVect, long lCount)

clDSPOp::Div1x

- void `dsp_div1x` (double *dpVect, long lCount)
- void `dsp_div1xf_nip` (float *fpDest, const float *fpSrc, long lCount)
- void `dsp_div1x_nip` (double *dpDest, const double *dpSrc, long lCount)
- void `dsp_muladdf` (float *fpVect, float fMul, float fAdd, long lCount)

clDSPOp::MulAdd

- void `dsp_muladd` (double *dpVect, double dMul, double dAdd, long lCount)
- void `dsp_muladdf_nip` (float *fpDest, const float *fpSrc, float fMul, float fAdd, long lCount)
- void `dsp_muladd_nip` (double *dpDest, const double *dpSrc, double dMul, double dAdd, long lCount)
- void `dsp_absf` (float *fpVect, long lCount)

clDSPOp::Abs

- void `dsp_abs` (double *dpVect, long lCount)
- void `dsp_absf_nip` (float *fpDest, const float *fpSrc, long lCount)
- void `dsp_abs_nip` (double *dpDest, const double *dpSrc, long lCount)
- void `dsp_sqrtf` (float *fpVect, long lCount)

clDSPOp::Sqrt

- void `dsp_sqrt` (double *dpVect, long lCount)
- void `dsp_sqrtf_nip` (float *fpDest, const float *fpSrc, long lCount)

- void `dsp_sqrt_nip` (double *dpDest, const double *dpSrc, long lCount)
- void `dsp_zerof` (float *fpVect, long lCount)
cdSPOp::Zero
- void `dsp_zero` (double *dpVect, long lCount)
- void `dsp_czerof` (stpSCplx spVect, long lCount)
- void `dsp_czero` (stpDCplx spVect, long lCount)
- void `dsp_setf` (float *fpDest, float fSrc, long lCount)
cdSPOp::Set
- void `dsp_set` (double *dpDest, double dSrc, long lCount)
- void `dsp_csetf` (stpSCplx spDest, stSCplx sSrc, long lCount)
- void `dsp_cset` (stpDCplx spDest, stDCplx sSrc, long lCount)
- void `dsp_set2f` (float *fpDest, float fSrc, long lStart, long lCount, long lLength)
- void `dsp_set2` (double *dpDest, double dSrc, long lStart, long lCount, long lLength)
- void `dsp_cset2f` (stpSCplx spDest, stSCplx sSrc, long lStart, long lCount, long lLength)
- void `dsp_cset2` (stpDCplx spDest, stDCplx sSrc, long lStart, long lCount, long lLength)
- void `dsp_clipf` (float *fpVect, float fValue, long lCount)
cdSPOp::Clip
- void `dsp_clip` (double *dpVect, double dValue, long lCount)
- void `dsp_clipf_nip` (float *fpDest, const float *fpSrc, float fValue, long lCount)
- void `dsp_clip_nip` (double *dpDest, const double *dpSrc, double dValue, long lCount)
- void `dsp_clip2f` (float *fpVect, float fMin, float fMax, long lCount)
- void `dsp_clip2` (double *dpVect, double dMin, double dMax, long lCount)
- void `dsp_clip2f_nip` (float *fpDest, const float *fpSrc, float fMin, float fMax, long lCount)
- void `dsp_clip2_nip` (double *dpDest, const double *dpSrc, double dMin, double dMax, long lCount)
- void `dsp_clipzerof` (float *fpVect, long lCount)
cdSPOp::ClipZero
- void `dsp_clipzero` (double *dpVect, long lCount)
- void `dsp_clipzerof_nip` (float *fpDest, const float *fpSrc, long lCount)
- void `dsp_clipzero_nip` (double *dpDest, const double *dpSrc, long lCount)
- void `dsp_copyf` (float *fpDest, const float *fpSrc, long lCount)
cdSPOp::Copy
- void `dsp_copy` (double *dpDest, const double *dpSrc, long lCount)
- void `dsp_ccopyf` (stpSCplx spDest, const stSCplx spSrc, long lCount)
- void `dsp_ccopy` (stpDCplx spDest, const stDCplx spSrc, long lCount)
- float `dsp_convolvef` (const float *fpSrc1, const float *fpSrc2, long lCount)
cdSPOp::Convolve

- double [dsp_convolve](#) (const double *dpSrc1, const double *dpSrc2, long lCount)
- void [dsp_convolve2f](#) (float *fpDest, const float *fpSrc1, const float *fpSrc2, long lCount)
- void [dsp_convolve2](#) (double *dpDest, const double *dpSrc1, const double *dpSrc2, long lCount)
- float [dsp_correlatef](#) (const float *fpSrc1, const float *fpSrc2, long lCount)
clDSPOp::Correlate
- double [dsp_correlate](#) (const double *dpSrc1, const double *dpSrc2, long lCount)
- void [dsp_correlate2f](#) (float *fpDest, const float *fpSrc1, const float *fpSrc2, long lCount)
- void [dsp_correlate2](#) (double *dpDest, const double *dpSrc1, const double *dpSrc2, long lCount)
- float [dsp_autocorrf](#) (const float *fpSrc, long lCount)
clDSPOp::AutoCorrelate
- double [dsp_autocorr](#) (const double *dpSrc, long lCount)
- void [dsp_autocorr2f](#) (float *fpDest, const float *fpSrc, long lCount)
- void [dsp_autocorr2](#) (double *dpDest, const double *dpSrc, long lCount)
- float [dsp_dotproductf](#) (const float *fpSrc1, const float *fpSrc2, long lCount)
clDSPOp::DotProduct
- double [dsp_dotproduct](#) (const double *dpSrc1, const double *dpSrc2, long lCount)
- void [dsp_minmaxf](#) (float *fpMin, float *fpMax, const float *fpSrc, long lCount)
clDSPOp::MinMax
- void [dsp_minmax](#) (double *dpMin, double *dpMax, const double *dpSrc, long lCount)
- float [dsp_meanf](#) (const float *fpSrc, long lCount)
clDSPOp::Mean
- double [dsp_mean](#) (const double *dpSrc, long lCount)
- float [dsp_medianf](#) (const float *fpSrc, long lCount)
- double [dsp_median](#) (const double *dpSrc, long lCount)
- void [dsp_negatef](#) (float *fpVect, long lCount)
clDSPOp::Negate
- void [dsp_negate](#) (double *dpVect, long lCount)
- void [dsp_negatef_nip](#) (float *fpDest, const float *fpSrc, long lCount)
- void [dsp_negate_nip](#) (double *dpDest, const double *dpSrc, long lCount)
- void [dsp_normalizef](#) (float *fpVect, long lCount)
clDSPOp::Normalize
- void [dsp_normalize](#) (double *dpVect, long lCount)

- void [dsp_normalizef_nip](#) (float *fpDest, const float *fpSrc, long lCount)
- void [dsp_normalize_nip](#) (double *dpDest, const double *dpSrc, long lCount)
- float [dsp_productf](#) (const float *fpSrc, long lCount)
clDSPOp::Product
- double [dsp_product](#) (const double *dpSrc, long lCount)
- void [dsp_reversef](#) (float *fpVect, long lCount)
clDSPOp::Reverse
- void [dsp_reverse](#) (double *dpVect, long lCount)
- void [dsp_reversef_nip](#) (float *fpDest, const float *fpSrc, long lCount)
- void [dsp_reverse_nip](#) (double *dpDest, const double *dpSrc, long lCount)
- void [dsp_scalef](#) (float *fpVect, long lCount)
clDSPOp::Scale
- void [dsp_scale](#) (double *dpVect, long lCount)
- void [dsp_scalef_nip](#) (float *fpDest, const float *fpSrc, long lCount)
- void [dsp_scale_nip](#) (double *dpDest, const double *dpSrc, long lCount)
- void [dsp_scale01f](#) (float *fpVect, long lCount)
clDSPOp::Scale01
- void [dsp_scale01](#) (double *dpVect, long lCount)
- void [dsp_scale01f_nip](#) (float *fpDest, const float *fpSrc, long lCount)
- void [dsp_scale01_nip](#) (double *dpDest, const double *dpSrc, long lCount)
- void [dsp_sortf](#) (float *fpVect, long lCount)
clDSPOp::Sort
- void [dsp_sort](#) (double *dpVect, long lCount)
- void [dsp_sortl](#) (long *lpVect, long lCount)
- void [dsp_stddevf](#) (float *fpStdDev, float *fpMean, const float *fpSrc, long lCount)
clDSPOp::StdDev
- void [dsp_stddev](#) (double *dpStdDev, double *dpMean, const double *dpSrc, long lCount)
- float [dsp_sumf](#) (const float *fpSrc, long lCount)
clDSPOp::Sum
- double [dsp_sum](#) (const double *dpSrc, long lCount)
- void [dsp_squaref](#) (float *fpVect, long lCount)
clDSPOp::Square
- void [dsp_square](#) (double *dpVect, long lCount)
- void [dsp_squaref_nip](#) (float *fpDest, const float *fpSrc, long lCount)
- void [dsp_square_nip](#) (double *dpDest, const double *dpSrc, long lCount)
- void [dsp_convertu8f](#) (float *fpDest, const unsigned char *ucpSrc, long lCount)

clDSPOp::Convert

- void [dsp_convertu8](#) (double *dpDest, const unsigned char *ucpSrc, long lCount)
- void [dsp_converts16f](#) (float *fpDest, const signed short *ipSrc, long lCount, int i12bit)
- void [dsp_converts16](#) (double *dpDest, const signed short *ipSrc, long lCount, int i12bit)
- void [dsp_converts32f](#) (float *fpDest, const signed int *ipSrc, long lCount, int i24bit)
- void [dsp_converts32](#) (double *dpDest, const signed int *ipSrc, long lCount, int i24bit)
- void [dsp_convertd64f](#) (float *fpDest, const double *dpSrc, long lCount)
- void [dsp_convertf32](#) (double *dpDest, const float *fpSrc, long lCount)
- void [dsp_convertf32c](#) (unsigned char *ucpDest, const float *fpSrc, long lCount)
- void [dsp_convertd64c](#) (unsigned char *ucpDest, const double *dpSrc, long lCount)
- void [dsp_convertf32s](#) (signed short *ipDest, const float *fpSrc, long lCount, int i12bit)
- void [dsp_convertd64s](#) (signed short *ipDest, const double *dpSrc, long lCount, int i12bit)
- void [dsp_convertf32i](#) (signed int *ipDest, const float *fpSrc, long lCount, int i24bit)
- void [dsp_convertd64i](#) (signed int *ipDest, const double *dpSrc, long lCount, int i24bit)
- void [dsp_cart2polarf](#) (float *fpMagn, float *fpPhase, const float *fpReal, const float *fpImag, long lCount)

clDSPOp::CartToPolar

- void [dsp_cart2polar](#) (double *dpMagn, double *dpPhase, const double *dpReal, const double *dpImag, long lCount)
- void [dsp_cart2polar2f](#) (float *fpMagn, float *fpPhase, const [stpSCplx](#) spCart, long lCount)
- void [dsp_cart2polar2](#) (double *dpMagn, double *dpPhase, const [stpDCplx](#) spCart, long lCount)
- void [dsp_cart2polar3f](#) ([stpSPolar](#) spPolar, const [stpSCplx](#) spCart, long lCount)
- void [dsp_cart2polar3](#) ([stpDPolar](#) spPolar, const [stpDCplx](#) spCart, long lCount)
- void [dsp_cart2polar4f](#) ([utpSCoord](#) upCoord, long lCount)
- void [dsp_cart2polar4](#) ([utpDCoord](#) upCoord, long lCount)
- void [dsp_polar2cartf](#) (float *fpReal, float *fpImag, const float *fpMagn, const float *fpPhase, long lCount)

clDSPOp::PolarToCart

- void [dsp_polar2cart](#) (double *dpReal, double *dpImag, const double *dpMagn, const double *dpPhase, long lCount)
- void [dsp_polar2cart2f](#) ([stpSCplx](#) spCart, const float *fpMagn, const float *fpPhase, long lCount)
- void [dsp_polar2cart2](#) ([stpDCplx](#) spCart, const double *dpMagn, const double *dpPhase, long lCount)

- void `dsp_polar2cart3f` (`stpSCplx` spCart, const `stpSPolar` spPolar, long lCount)
- void `dsp_polar2cart3` (`stpDCplx` spCart, const `stpDPolar` spPolar, long lCount)
- void `dsp_polar2cart4f` (`utpSCoord` upCoord, long lCount)
- void `dsp_polar2cart4` (`utpDCoord` upCoord, long lCount)
- float `dsp_crosscorrff` (const float *fpSrc1, const float *fpSrc2, long lCount)
clDSPOp::CrossCorr
- double `dsp_crosscorr` (const double *dpSrc1, const double *dpSrc2, long lCount)
- float `dsp_crosscorr2f` (const float *fpSrc1, const float *fpSrc2, long lDelay, long lCount)
clDSPOp::DelCrossCorr
- double `dsp_crosscorr2` (const double *dpSrc1, const double *dpSrc2, long lDelay, long lCount)
- void `dsp_crosscorr3f` (float *fpDest, const float *fpSrc1, const float *fpSrc2, long lCount, const long *lpDelays, long lDelayCount)
- void `dsp_crosscorr3` (double *dpDest, const double *dpSrc1, const double *dpSrc2, long lCount, const long *lpDelays, long lDelayCount)
- float `dsp_energyf` (const float *fpSrc, long lCount)
clDSPOp::Energy
- double `dsp_energy` (const double *dpSrc, long lCount)
- void `dsp_magnitudef` (float *fpDest, const `stpSCplx` spSrc, long lCount)
clDSPOp::Magnitude
- void `dsp_magnitude` (double *dpDest, const `stpDCplx` spSrc, long lCount)
- void `dsp_powerf` (float *fpDest, const `stpSCplx` spSrc, long lCount)
clDSPOp::Power
- void `dsp_power` (double *dpDest, const `stpDCplx` spSrc, long lCount)
- void `dsp_phasef` (float *fpDest, const `stpSCplx` spSrc, long lCount)
clDSPOp::Phase
- void `dsp_phase` (double *dpDest, const `stpDCplx` spSrc, long lCount)
- void `dsp_powerphasef` (float *fpPower, float *fpPhase, const `stpSCplx` spSrc, long lCount)
clDSPOp::PowerPhase
- void `dsp_powerphase` (double *dpPower, double *dpPhase, const `stpDCplx` spSrc, long lCount)
- void `dsp_decimatef` (float *fpDest, const float *fpSrc, long lFactor, long lCount)
clDSPOp::Decimate
- void `dsp_decimate` (double *dpDest, const double *dpSrc, long lFactor, long lCount)

- void [dsp_decimateavgf](#) (float *fpDest, const float *fpSrc, long lFactor, long lCount)
clDSPOp::DecimateAvg
- void [dsp_decimateavg](#) (double *dpDest, const double *dpSrc, long lFactor, long lCount)
- void [dsp_interpolatef](#) (float *fpDest, const float *fpSrc, long lFactor, long lCount)
clDSPOp::Interpolate
- void [dsp_interpolate](#) (double *dpDest, const double *dpSrc, long lFactor, long lCount)
- void [dsp_resamplef](#) (float *fpDest, long lDestCount, const float *fpSrc, long lSrcCount)
clDSPOp::Resample
- void [dsp_resample](#) (double *dpDest, long lDestCount, const double *dpSrc, long lSrcCount)
- void [dsp_resampleavgf](#) (float *fpDest, long lDestCount, const float *fpSrc, long lSrcCount)
clDSPOp::ResampleAvg
- void [dsp_resampleavg](#) (double *dpDest, long lDestCount, const double *dpSrc, long lSrcCount)
- void [dsp_interpolateavgf](#) (float *fpDest, const float *fpSrc, long lFactor, long lCount)
clDSPOp::InterpolateAvg
- void [dsp_interpolateavg](#) (double *dpDest, const double *dpSrc, long lFactor, long lCount)
- float [dsp_rmsf](#) (const float *fpSrc, long lCount)
clDSPOp::RMS
- double [dsp_rms](#) (const double *dpSrc, long lCount)
- float [dsp_variancef](#) (float *fpVariance, float *fpMean, const float *fpSrc, long lCount)
clDSPOp::Variance
- double [dsp_variance](#) (double *dpVariance, double *dpMean, const double *dpSrc, long lCount)
- float [dsp_peaklevelf](#) (const float *fpSrc, long lCount)
clDSPOp::PeakLevel
- double [dsp_peaklevel](#) (const double *dpSrc, long lCount)
- void [dsp_mixf](#) (float *fpDest, const float *fpSrc, long lCount)
clDSPOp::Mix

- void [dsp_mix](#) (double *dpDest, const double *dpSrc, long lCount)
- void [dsp_mix2f](#) (float *fpDest, const float *fpSrc1, const float *fpSrc2, long lCount)
- void [dsp_mix2](#) (double *dpDest, const double *dpSrc1, const double *dpSrc2, long lCount)
- void [dsp_mixnf](#) (float *fpDest, const float *fpSrc, long lChCount, long lCount)
- void [dsp_mixn](#) (double *dpDest, const double *dpSrc, long lChCount, long lCount)
- void [dsp_extractf](#) (float *fpDest, const float *fpSrc, long lChannel, long lChCount, long lCount)
clDSPOp::Extract
- void [dsp_extract](#) (double *dpDest, const double *dpSrc, long lChannel, long lChCount, long lCount)
- void [dsp_packf](#) (float *fpDest, const float *fpSrc, long lChannel, long lChCount, long lCount)
clDSPOp::Pack
- void [dsp_pack](#) (double *dpDest, const double *dpSrc, long lChannel, long lChCount, long lCount)
- void [dsp_fftw_convertf2cf](#) (stpSCplx spDest, const float *fpSrc, long lCount)
clDSPOp::FFTWConvert
- void [dsp_fftw_convertf2cd](#) (stpDCplx spDest, const float *fpSrc, long lCount)
- void [dsp_fftw_convertd2cf](#) (stpSCplx spDest, const double *dpSrc, long lCount)
- void [dsp_fftw_convertd2cd](#) (stpDCplx spDest, const double *dpSrc, long lCount)
- void [dsp_fftw_convertcf2f](#) (float *fpDest, const stpSCplx spSrc, long lCount)
- void [dsp_fftw_convertcd2f](#) (float *fpDest, const stpDCplx spSrc, long lCount)
- void [dsp_fftw_convertcf2d](#) (double *dpDest, const stpSCplx spSrc, long lCount)
- void [dsp_fftw_convertcd2d](#) (double *dpDest, const stpDCplx spSrc, long lCount)
- [dsp_t dsp_new](#) ()
Creates new instance of DSP object.
- void [dsp_delete](#) (dsp_t dspInst)
Deletes DSP object instance.
- void [dsp_win_bartlett](#) (dsp_t dspInst, float *fpDest, long lCount)
clDSPOp::WinBartlett
- void [dsp_win_bartlett](#) (dsp_t dspInst, double *dpDest, long lCount)
- void [dsp_win_blackmanf](#) (dsp_t dspInst, float *fpDest, long lCount, float fAlpha)
clDSPOp::WinBlackman

- void `dsp_win_blackman` (`dsp_t` dspInst, double *dpDest, long lCount, double dAlpha)
- void `dsp_win_blackman_harris` (`dsp_t` dspInst, float *fpDest, long lCount)
clDSPOp::WinBlackmanHarris
- void `dsp_win_blackman_harris` (`dsp_t` dspInst, double *dpDest, long lCount)
- void `dsp_win_cos_taperedf` (`dsp_t` dspInst, float *fpDest, long lCount)
clDSPOp::WinCosTapered
- void `dsp_win_cos_tapered` (`dsp_t` dspInst, double *dpDest, long lCount)
- void `dsp_win_exact_blackmanf` (`dsp_t` dspInst, float *fpDest, long lCount)
clDSPOp::WinExactBlackman
- void `dsp_win_exact_blackman` (`dsp_t` dspInst, double *dpDest, long lCount)
- void `dsp_win_flat_topf` (`dsp_t` dspInst, float *fpDest, long lCount)
clDSPOp::WinFlatTop
- void `dsp_win_flat_top` (`dsp_t` dspInst, double *dpDest, long lCount)
- void `dsp_win_generic_cosf` (`dsp_t` dspInst, float *fpDest, long lCount, const float *fpCoeff, long lCoeffCount)
clDSPOp::WinGenericCos
- void `dsp_win_generic_cos` (`dsp_t` dspInst, double *dpDest, long lCount, const double *dpCoeff, long lCoeffCount)
- void `dsp_win_hammingf` (`dsp_t` dspInst, float *fpDest, long lCount)
clDSPOp::WinHamming
- void `dsp_win_hamming` (`dsp_t` dspInst, double *dpDest, long lCount)
- void `dsp_win_hanningf` (`dsp_t` dspInst, float *fpDest, long lCount)
clDSPOp::WinHanning
- void `dsp_win_hanning` (`dsp_t` dspInst, double *dpDest, long lCount)
- void `dsp_win_kaiserf` (`dsp_t` dspInst, float *fpDest, float fBeta, long lCount)
clDSPOp::WinKaiser
- void `dsp_win_kaiser` (`dsp_t` dspInst, double *dpDest, double dBeta, long lCount)
- void `dsp_win_kaiser_besself` (`dsp_t` dspInst, float *fpDest, float fAlpha, long lCount)
clDSPOp::WinKaiserBessel
- void `dsp_win_kaiser_bessel` (`dsp_t` dspInst, double *dpDest, double dAlpha, long lCount)
- void `dsp_win_tukeyf` (`dsp_t` dspInst, float *fpDest, long lCount)
clDSPOp::WinTukey
- void `dsp_win_tukey` (`dsp_t` dspInst, double *dpDest, long lCount)

- void `dsp_win_dolph_chebyshev` (`dsp_t` dspInst, float *fpDest, float fGamma, long lCount)
cdSPOp::WinDolphChebyshev
- void `dsp_win_dolph_chebyshev` (`dsp_t` dspInst, double *dpDest, double dGamma, long lCount)
- long `dsp_rebufferf` (`dsp_t` dspInst, float *fpDest, const float *fpSrc, long lDestCount, long lSrcCount)
cdSPOp::ReBuffer
- long `dsp_rebuffer` (`dsp_t` dspInst, double *dpDest, const double *dpSrc, long lDestCount, long lSrcCount)
- float `dsp_deg2radf` (`dsp_t` dspInst, float fSource)
cdSPOp::DegToRad
- double `dsp_deg2rad` (`dsp_t` dspInst, double dSource)
- float `dsp_rad2degf` (`dsp_t` dspInst, float fSource)
cdSPOp::RadToDeg
- double `dsp_rad2deg` (`dsp_t` dspInst, double dSource)
- void `dsp_fir_allocatef` (`dsp_t` dspInst, const float *fpCoeff, long lCount)
cdSPOp::FIRAllocate
- void `dsp_fir_allocate` (`dsp_t` dspInst, const double *dpCoeff, long lCount)
- void `dsp_fir_free` (`dsp_t` dspInst)
cdSPOp::FIRFree
- void `dsp_fir_filterf` (`dsp_t` dspInst, float *fpVect, long lCount)
cdSPOp::FIRFilter
- void `dsp_fir_filter` (`dsp_t` dspInst, double *dpVect, long lCount)
- void `dsp_fir_filterf_nip` (`dsp_t` dspInst, float *fpDest, const float *fpSrc, long lCount)
- void `dsp_fir_filter_nip` (`dsp_t` dspInst, double *dpDest, const double *dpSrc, long lCount)
- void `dsp_fir_filterf_fst` (`dsp_t` dspInst, float *fpDest, float *fpSrc, long lCount)
cdSPOp::FIRFilterF
- void `dsp_fir_filter_fst` (`dsp_t` dspInst, double *dpDest, double *dpSrc, long lCount)
- void `dsp_iir_initf` (`dsp_t` dspInst, const float *fpCoeffs)
cdSPOp::IIRInitialize
- void `dsp_iir_init` (`dsp_t` dspInst, const double *dpCoeffs)
- void `dsp_iir_filterf` (`dsp_t` dspInst, float *fpVect, long lCount)
cdSPOp::IIRFilter

- void `dsp_iir_filter` (`dsp_t` dspInst, double *dpVect, long lCount)
- void `dsp_iir_filterf_nip` (`dsp_t` dspInst, float *fpDest, const float *fpSrc, long lCount)
- void `dsp_iir_filter_nip` (`dsp_t` dspInst, double *dpDest, const double *dpSrc, long lCount)
- void `dsp_iir_clear` (`dsp_t` dspInst)
clDSPOp::IIRClear
- void `dsp_fft_init` (`dsp_t` dspInst, long lSize, int iReal)
clDSPOp::FFTInitialize
- void `dsp_fft_uninit` (`dsp_t` dspInst)
clDSPOp::FFTUninitialize
- void `dsp_fftf` (`dsp_t` dspInst, `stpSCplx` spDest, float *fpSrc)
clDSPOp::FFTi
- void `dsp_fft` (`dsp_t` dspInst, `stpDCplx` spDest, double *dpSrc)
- void `dsp_fftf_nip` (`dsp_t` dspInst, `stpSCplx` spDest, const float *fpSrc)
clDSPOp::FFTo
- void `dsp_fft_nip` (`dsp_t` dspInst, `stpDCplx` spDest, const double *dpSrc)
- void `dsp_cfft_nip` (`dsp_t` dspInst, `stpSCplx` spDest, const `stpSCplx` spSrc)
- void `dsp_cfft_nip` (`dsp_t` dspInst, `stpDCplx` spDest, const `stpDCplx` spSrc)
- void `dsp_iftf_nip` (`dsp_t` dspInst, float *fpDest, const `stpSCplx` spSrc)
clDSPOp::IFFTo
- void `dsp_iftf_nip` (`dsp_t` dspInst, double *dpDest, const `stpDCplx` spSrc)
- void `dsp_ciftf_nip` (`dsp_t` dspInst, `stpSCplx` spDest, const `stpSCplx` spSrc)
- void `dsp_ciftf_nip` (`dsp_t` dspInst, `stpDCplx` spDest, const `stpDCplx` spSrc)
- `dsp_iircas_t` `dsp_iir_cas_new` ()
Creates new instance of cascaded IIR object.
- void `dsp_iir_cas_delete` (`dsp_iircas_t` dspiircasInst)
Deletes cascaded IIR object instance.
- int `dsp_iir_cas_initf` (`dsp_iircas_t` dspiircasInst, const float fpCoeffs[][5], long lStageCount)
clIIRCascade::Initialize
- int `dsp_iir_cas_init` (`dsp_iircas_t` dspiircasInst, const double dpCoeffs[][5], long lStageCount)
- void `dsp_iir_cas_uninit` (`dsp_iircas_t` dspiircasInst)
- void `dsp_iir_cas_processf` (`dsp_iircas_t` dspiircasInst, float *fpVect, long lCount)
- void `dsp_iir_cas_process` (`dsp_iircas_t` dspiircasInst, double *dpVect, long lCount)

- void [dsp_iir_cas_processf_nip](#) ([dsp_iircas_t](#) dspiircasInst, float *fpDest, const float *fpSrc, long lCount)
- void [dsp_iir_cas_process_nip](#) ([dsp_iircas_t](#) dspiircasInst, double *dpDest, const double *dpSrc, long lCount)
- void [dsp_iir_cas_clear](#) ([dsp_iircas_t](#) dspiircasInst)
- [dsp_decfft_t](#) [dsp_dec_fft_new](#) ()
Creates new instance of FFT decimator object.
- void [dsp_dec_fft_delete](#) ([dsp_decfft_t](#) dspdecfftInst)
Deletes FFT decimator object instance.
- int [dsp_dec_fft_initf](#) ([dsp_decfft_t](#) dspdecfftInst, long lDecFact, long lFiltSize, int iHighPass)
clFFTDecimator::Initialize
- int [dsp_dec_fft_init](#) ([dsp_decfft_t](#) dspdecfftInst, long lDecFact, long lFiltSize, int iHighPass)
- void [dsp_dec_fft_uninit](#) ([dsp_decfft_t](#) dspdecfftInst)
clFFTDecimator::Uninitialize
- void [dsp_dec_fft_putf](#) ([dsp_decfft_t](#) dspdecfftInst, const float *fpSrcData, long lSrcCount)
clFFTDecimator::Put
- void [dsp_dec_fft_put](#) ([dsp_decfft_t](#) dspdecfftInst, const double *dpSrcData, long lSrcCount)
- int [dsp_dec_fft_getf](#) ([dsp_decfft_t](#) dspdecfftInst, float *fpDestData, long lDestCount)
clFFTDecimator::Get
- int [dsp_dec_fft_get](#) ([dsp_decfft_t](#) dspdecfftInst, double *dpDestData, long lDestCount)
- [dsp_decfir_t](#) [dsp_dec_fir_new](#) ()
Creates new instance of FIR decimator object.
- void [dsp_dec_fir_delete](#) ([dsp_decfir_t](#) dspdecfirInst)
Deletes FIR decimator object instance.
- int [dsp_dec_fir_initf](#) ([dsp_decfir_t](#) dspdecfirInst, long lDecFact, int iHighPass)
clFIRDecimator::Initialize
- int [dsp_dec_fir_init](#) ([dsp_decfir_t](#) dspdecfirInst, long lDecFact, int iHighPass)
- void [dsp_dec_fir_uninit](#) ([dsp_decfir_t](#) dspdecfirInst)
clFIRDecimator::Uninitialize
- void [dsp_dec_fir_putf](#) ([dsp_decfir_t](#) dspdecfirInst, const float *fpSrcData, long lSrcCount)

clFIRDecimator::Put

- void `dsp_dec_fir_put` (`dsp_decfir_t` dspdecfirInst, const double *dpSrcData, long lSrcCount)
- int `dsp_dec_fir_getf` (`dsp_decfir_t` dspdecfirInst, float *fpDestData, long lDestCount)

clFIRDecimator::Get

- int `dsp_dec_fir_get` (`dsp_decfir_t` dspdecfirInst, double *dpDestData, long lDestCount)
- `dsp_deciir_t dsp_dec_iir_new` ()

Creates new instance of IIR decimator object.

- void `dsp_dec_iir_delete` (`dsp_deciir_t` dspdeciirInst)

Deletes IIR decimator object instance.

- int `dsp_dec_iir_initf` (`dsp_deciir_t` dspdeciirInst, long lDecFact, int iHighPass)

clIIRDecimator::Initialize

- int `dsp_dec_iir_init` (`dsp_deciir_t` dspdeciirInst, long lDecFact, int iHighPass)
- void `dsp_dec_iir_uninit` (`dsp_deciir_t` dspdeciirInst)

clIIRDecimator::Uninitialize

- void `dsp_dec_iir_putf` (`dsp_deciir_t` dspdeciirInst, const float *fpSrcData, long lSrcCount)

clIIRDecimator::Put

- void `dsp_dec_iir_put` (`dsp_deciir_t` dspdeciirInst, const double *dpSrcData, long lSrcCount)
- int `dsp_dec_iir_getf` (`dsp_deciir_t` dspdeciirInst, float *fpDestData, long lDestCount)

clIIRDecimator::Get

- int `dsp_dec_iir_get` (`dsp_deciir_t` dspdeciirInst, double *dpDestData, long lDestCount)
- `dsp_decrec_t dsp_dec_rec_new` ()

Creates new instance of recursive decimator object.

- void `dsp_dec_rec_delete` (`dsp_decrec_t` dspdecrecInst)

Deletes recursive decimator object instance.

- int `dsp_dec_rec_initf` (`dsp_decrec_t` dspdecrecInst, long lDecFact, long lFiltSize, float fBandCenter, int iFilterType)

clRecDecimator::Initialize

- int `dsp_dec_rec_init` (`dsp_decrec_t` dspdecrecInst, long lDecFact, long lFiltSize, double dBandCenter, int iFilterType)

- void `dsp_dec_rec_uninit` (`dsp_decrec_t` dspdecrecInst)
clRecDecimator::Uninitialize
- void `dsp_dec_rec_putf` (`dsp_decrec_t` dspdecrecInst, const float *fpSrcData, long lSrcCount)
clRecDecimator::Put
- void `dsp_dec_rec_put` (`dsp_decrec_t` dspdecrecInst, const double *dpSrcData, long lSrcCount)
- int `dsp_dec_rec_getf` (`dsp_decrec_t` dspdecrecInst, float *fpDestData, long lDestCount)
clRecDecimator::Get
- int `dsp_dec_rec_get` (`dsp_decrec_t` dspdecrecInst, double *dpDestData, long lDestCount)
- `dsp_intfft_t` `dsp_int_fft_new` ()
Creates new instance of FFT interpolator object.
- void `dsp_int_fft_delete` (`dsp_intfft_t` dspintfftInst)
Deletes FFT interpolator object instance.
- int `dsp_int_fft_initf` (`dsp_intfft_t` dspintfftInst, long lIntFact, long lFiltSize, int iHighPass)
clFFTInterpolator::Initialize
- int `dsp_int_fft_init` (`dsp_intfft_t` dspintfftInst, long lIntFact, long lFiltSize, int iHighPass)
- void `dsp_int_fft_uninit` (`dsp_intfft_t` dspintfftInst)
clFFTInterpolator::Uninitialize
- void `dsp_int_fft_putf` (`dsp_intfft_t` dspintfftInst, const float *fpSrcData, long lSrcCount)
clFFTInterpolator::Put
- void `dsp_int_fft_put` (`dsp_intfft_t` dspintfftInst, const double *dpSrcData, long lSrcCount)
- int `dsp_int_fft_getf` (`dsp_intfft_t` dspintfftInst, float *fpDestData, long lDestCount)
clFFTInterpolator::Get
- int `dsp_int_fft_get` (`dsp_intfft_t` dspintfftInst, double *dpDestData, long lDestCount)
- `dsp_intfir_t` `dsp_int_fir_new` ()
Creates new instance of FIR interpolator object.
- void `dsp_int_fir_delete` (`dsp_intfir_t` dspintfirInst)
Deletes FIR interpolator object instance.

- int `dsp_int_fir_initf` (`dsp_intfir_t` dspintfirInst, long lIntFact, int iHighPass)
clFIRInterpolator::Initialize
- int `dsp_int_fir_init` (`dsp_intfir_t` dspintfirInst, long lIntFact, int iHighPass)
- void `dsp_int_fir_uninit` (`dsp_intfir_t` dspintfirInst)
clFIRInterpolator::Uninitialize
- void `dsp_int_fir_putf` (`dsp_intfir_t` dspintfirInst, const float *fpSrcData, long lSrcCount)
clFIRInterpolator::Put
- void `dsp_int_fir_put` (`dsp_intfir_t` dspintfirInst, const double *dpSrcData, long lSrcCount)
- int `dsp_int_fir_getf` (`dsp_intfir_t` dspintfirInst, float *fpDestData, long lDestCount)
clFIRInterpolator::Get
- int `dsp_int_fir_get` (`dsp_intfir_t` dspintfirInst, double *dpDestData, long lDestCount)
- `dsp_intiir_t` `dsp_int_iir_new` ()
Creates new instance of IIR interpolator object.
- void `dsp_int_iir_delete` (`dsp_intiir_t` dspintiirInst)
Deletes IIR interpolator object instance.
- int `dsp_int_iir_initf` (`dsp_intiir_t` dspintiirInst, long lIntFact, int iHighPass)
clIIRInterpolator::Initialize
- int `dsp_int_iir_init` (`dsp_intiir_t` dspintiirInst, long lIntFact, int iHighPass)
- void `dsp_int_iir_uninit` (`dsp_intiir_t` dspintiirInst)
clIIRInterpolator::Uninitialize
- void `dsp_int_iir_putf` (`dsp_intiir_t` dspintiirInst, const float *fpSrcData, long lSrcCount)
clIIRInterpolator::Put
- void `dsp_int_iir_put` (`dsp_intiir_t` dspintiirInst, const double *dpSrcData, long lSrcCount)
- int `dsp_int_iir_getf` (`dsp_intiir_t` dspintiirInst, float *fpDestData, long lDestCount)
clIIRInterpolator::Get
- int `dsp_int_iir_get` (`dsp_intiir_t` dspintiirInst, double *dpDestData, long lDestCount)
- `dsp_intrec_t` `dsp_int_rec_new` ()
Creates new instance of recursive interpolator object.

- void [dsp_int_rec_delete](#) ([dsp_intrec_t](#) dspintrecInst)
Deletes recursive interpolator object instance.
- int [dsp_int_rec_initf](#) ([dsp_intrec_t](#) dspintrecInst, long lIntFact, long lFiltSize, float fBandCenter, int iFilterType)
clRecInterpolator::Initialize
- int [dsp_int_rec_init](#) ([dsp_intrec_t](#) dspintrecInst, long lIntFact, long lFiltSize, double dBandCenter, int iFilterType)
- void [dsp_int_rec_uninit](#) ([dsp_intrec_t](#) dspintrecInst)
clRecInterpolator::Uninitialize
- void [dsp_int_rec_putf](#) ([dsp_intrec_t](#) dspintrecInst, const float *fpSrcData, long lSrcCount)
clRecInterpolator::Put
- void [dsp_int_rec_put](#) ([dsp_intrec_t](#) dspintrecInst, const double *dpSrcData, long lSrcCount)
- int [dsp_int_rec_getf](#) ([dsp_intrec_t](#) dspintrecInst, float *fpDestData, long lDestCount)
clRecInterpolator::Get
- int [dsp_int_rec_get](#) ([dsp_intrec_t](#) dspintrecInst, double *dpDestData, long lDestCount)
- [dsp_filter_t](#) [dsp_filter_new](#) ()
Creates new instance of FFT filter object.
- void [dsp_filter_delete](#) ([dsp_filter_t](#) dspfilterInst)
Deletes instance of FFT filter object.
- int [dsp_filter_initf](#) ([dsp_filter_t](#) dspfilterInst, long lWindowSize)
clFilter::Initialize
- int [dsp_filter_init](#) ([dsp_filter_t](#) dspfilterInst, long lWindowSize)
- int [dsp_filter_init2f](#) ([dsp_filter_t](#) dspfilterInst, long lWindowSize, const float *fpFiltCoeffs, float fOverlap, float fBeta)
- int [dsp_filter_init2](#) ([dsp_filter_t](#) dspfilterInst, long lWindowSize, const double *dpFiltCoeffs, double dOverlap, double dBeta)
- int [dsp_filter_init_lpf](#) ([dsp_filter_t](#) dspfilterInst, float fPassBand, float fStopBand, float fRippleRatio, float fOverlap)
clFilter::InitializeLP
- int [dsp_filter_init_lp](#) ([dsp_filter_t](#) dspfilterInst, double dPassBand, double dStopBand, double dRippleRatio, double dOverlap)
- int [dsp_filter_init_hpf](#) ([dsp_filter_t](#) dspfilterInst, float fPassBand, float fStopBand, float fRippleRatio, float fOverlap)

clFilter::InitializeHP

- int `dsp_filter_init_hp` (`dsp_filter_t` dspfilterInst, double dPassBand, double d-StopBand, double dRippleRatio, double dOverlap)
- void `dsp_filter_set_coefssf` (`dsp_filter_t` dspfilterInst, const float *fpFiltCoeffs)

clFilter::SetCoeffs

- void `dsp_filter_set_coefss` (`dsp_filter_t` dspfilterInst, const double *dpFiltCoeffs)
- void `dsp_filter_set_ccoeffsf` (`dsp_filter_t` dspfilterInst, const `stpSCplx` spFiltCoeffs, int iSmooth)
- void `dsp_filter_set_ccoeffs` (`dsp_filter_t` dspfilterInst, const `stpDCplx` spFiltCoeffs, int iSmooth)
- void `dsp_filter_get_coefssf` (`dsp_filter_t` dspfilterInst, float *fpFiltCoeffs)

clFilter::GetCoeffs

- void `dsp_filter_get_coefss` (`dsp_filter_t` dspfilterInst, double *dpFiltCoeffs)
- void `dsp_filter_get_ccoeffsf` (`dsp_filter_t` dspfilterInst, `stpSCplx` spFiltCoeffs)
- void `dsp_filter_get_ccoeffs` (`dsp_filter_t` dspfilterInst, `stpDCplx` spFiltCoeffs)
- void `dsp_filter_putf` (`dsp_filter_t` dspfilterInst, const float *fpSrcData, long lSrcCount)

clFilter::Put

- void `dsp_filter_put` (`dsp_filter_t` dspfilterInst, const double *dpSrcData, long lSrcCount)
- void `dsp_filter_put2f` (`dsp_filter_t` dspfilterInst, const float *fpSrcData, long lSrcCount, const `stpSCplx` spCoeffs)
- void `dsp_filter_put2` (`dsp_filter_t` dspfilterInst, const double *dpSrcData, long lSrcCount, const `stpDCplx` spCoeffs)
- void `dsp_filter_getf` (`dsp_filter_t` dspfilterInst, float *fpDestData, long lDestCount)

clFilter::Get

- void `dsp_filter_get` (`dsp_filter_t` dspfilterInst, double *dpDestData, long lDestCount)
- void `dsp_filter_design_lpf` (`dsp_filter_t` dspfilterInst, float *fpCorner, int iDCBlock)

clFilter::DesignLP

- void `dsp_filter_design_lp` (`dsp_filter_t` dspfilterInst, double *dpCorner, int iDCBlock)
- void `dsp_filter_design_lp2f` (`dsp_filter_t` dspfilterInst, float *fpCorner, float f-SampleRate, int iDCBlock)
- void `dsp_filter_design_lp2` (`dsp_filter_t` dspfilterInst, double *dpCorner, double dSampleRate, int iDCBlock)
- void `dsp_filter_design_hpf` (`dsp_filter_t` dspfilterInst, float *fpCorner)

clFilter::DesignHP

- void [dsp_filter_design_hp](#) ([dsp_filter_t](#) dspfilterInst, double *dpCorner)
- void [dsp_filter_design_hp2f](#) ([dsp_filter_t](#) dspfilterInst, float *fpCorner, float fSampleRate)
- void [dsp_filter_design_hp2](#) ([dsp_filter_t](#) dspfilterInst, double *dpCorner, double dSampleRate)
- void [dsp_filter_design_bpf](#) ([dsp_filter_t](#) dspfilterInst, float *fpLowCorner, float *fpHighCorner)

clFilter::DesignBP

- void [dsp_filter_design_bp](#) ([dsp_filter_t](#) dspfilterInst, double *dpLowCorner, double *dpHighCorner)
- void [dsp_filter_design_bp2f](#) ([dsp_filter_t](#) dspfilterInst, float *fpLowCorner, float *fpHighCorner, float fSampleRate)
- void [dsp_filter_design_bp2](#) ([dsp_filter_t](#) dspfilterInst, double *dpLowCorner, double *dpHighCorner, double dSampleRate)
- void [dsp_filter_design_brf](#) ([dsp_filter_t](#) dspfilterInst, float *fpLowCorner, float *fpHighCorner)

clFilter::DesignBR

- void [dsp_filter_design_br](#) ([dsp_filter_t](#) dspfilterInst, double *dpLowCorner, double *dpHighCorner)
- void [dsp_filter_design_br2f](#) ([dsp_filter_t](#) dspfilterInst, float *fpLowCorner, float *fpHighCorner, float fSampleRate)
- void [dsp_filter_design_br2](#) ([dsp_filter_t](#) dspfilterInst, double *dpLowCorner, double *dpHighCorner, double dSampleRate)
- [dsp_rebuf_t dsp_rebuf_new](#) ()

Creates new instance of rebuffering object.

- void [dsp_rebuf_delete](#) ([dsp_rebuf_t](#) dsprebufInst)

Deletes instance of rebuffering object.

- void [dsp_rebuf_putf](#) ([dsp_rebuf_t](#) dsprebufInst, const float *fpSrcData, long lSrcCount)

clReBuffer::Put

- void [dsp_rebuf_put](#) ([dsp_rebuf_t](#) dsprebufInst, const double *dpSrcData, long lSrcCount)
- int [dsp_rebuf_getf](#) ([dsp_rebuf_t](#) dsprebufInst, float *fpDestData, long lDestCount)

clReBuffer::Get

- int [dsp_rebuf_get](#) ([dsp_rebuf_t](#) dsprebufInst, double *dpDestData, long lDestCount)
- long [dsp_rebuf_size](#) ([dsp_rebuf_t](#) dsprebufInst)

clReBuffer::GetCount

- void [dsp_rebuf_clear](#) ([dsp_rebuf_t](#) dsprebufInst)

[*clReBuffer::Clear*](#)

- void [dsp_rebuf_copy](#) ([dsp_rebuf_t](#) dsprebufInst, [dsp_rebuf_t](#) dsprebufCopySrc)

[*clReBuffer::operator=*](#)

6.2.1 Function Documentation

6.2.1.1 void dsp_init (void)

Initialize DSP library.

This should be called first to enable possible hardware dependent optimizations.

6.2.1.2 signed long dsp_roundf (float fSrc)

[clDSPOp::Round](#)

6.2.1.3 signed long dsp_round (double dSrc)

6.2.1.4 void dsp_addf (float *fpDest, float fSrc, long lCount)

[clDSPOp::Add](#)

6.2.1.5 void dsp_add (double * *dpDest*, double *dSrc*, long *lCount*)

6.2.1.6 void dsp_caddf ([stSCplx](#) *spDest*, [stSCplx](#) *sSrc*, long *lCount*)

6.2.1.7 void dsp_cadd ([stDCplx](#) *spDest*, [stDCplx](#) *sSrc*, long *lCount*)

6.2.1.8 void dsp_add2f (float * *fpDest*, const float * *fpSrc*, long *lCount*)

6.2.1.9 void dsp_add2 (double * *dpDest*, const double * *dpSrc*, long *lCount*)

6.2.1.10 void dsp_cadd2f ([stSCplx](#) *spDest*, const [stSCplx](#) *spSrc*, long *lCount*)

6.2.1.11 void dsp_cadd2 ([stDCplx](#) *spDest*, const [stDCplx](#) *spSrc*, long *lCount*)

6.2.1.12 void dsp_add3f (float * *fpDest*, const float * *fpSrc1*, const float * *fpSrc2*, long *lCount*)

6.2.1.13 void dsp_add3 (double * *dpDest*, const double * *dpSrc1*, const double * *dpSrc2*, long *lCount*)

6.2.1.14 void dsp_cadd3f ([stSCplx](#) *spDest*, const [stSCplx](#) *spSrc1*, const [stSCplx](#) *spSrc2*, long *lCount*)

6.2.1.15 void dsp_cadd3 ([stDCplx](#) *spDest*, const [stDCplx](#) *spSrc1*, const [stDCplx](#) *spSrc2*, long *lCount*)

6.2.1.16 void dsp_subf (float * *fpDest*, float *fSrc*, long *lCount*)

[clDSPOp::Sub](#)

- 6.2.1.17 void dsp_sub (double * *dpDest*, double *dSrc*, long *lCount*)
- 6.2.1.18 void dsp_csubf ([stpSCplx](#) *spDest*, [stSCplx](#) *sSrc*, long *lCount*)
- 6.2.1.19 void dsp_csub ([stpDCplx](#) *spDest*, [stDCplx](#) *sSrc*, long *lCount*)
- 6.2.1.20 void dsp_sub2f (float * *fpDest*, const float * *fpSrc*, long *lCount*)
- 6.2.1.21 void dsp_sub2 (double * *dpDest*, const double * *dpSrc*, long *lCount*)
- 6.2.1.22 void dsp_csub2f ([stpSCplx](#) *spDest*, const [stpSCplx](#) *spSrc*, long *lCount*)
- 6.2.1.23 void dsp_csub2 ([stpDCplx](#) *spDest*, const [stpDCplx](#) *spSrc*, long *lCount*)
- 6.2.1.24 void dsp_sub3f (float * *fpDest*, const float * *fpSrc1*, const float * *fpSrc2*, long *lCount*)
- 6.2.1.25 void dsp_sub3 (double * *dpDest*, const double * *dpSrc1*, const double * *dpSrc2*, long *lCount*)
- 6.2.1.26 void dsp_csub3f ([stpSCplx](#) *spDest*, const [stpSCplx](#) *spSrc1*, const [stpSCplx](#) *spSrc2*, long *lCount*)
- 6.2.1.27 void dsp_csub3 ([stpDCplx](#) *spDest*, const [stpDCplx](#) *spSrc1*, const [stpDCplx](#) *spSrc2*, long *lCount*)
- 6.2.1.28 void dsp_mulf (float * *fpDest*, float *fSrc*, long *lCount*)

[clDSPOp::Mul](#)

- 6.2.1.29 void dsp_mul (double * *dpDest*, double *dSrc*, long *lCount*)
- 6.2.1.30 void dsp_chmulf ([stpSCplx](#) *spDest*, float *fSrc*, long *lCount*)
- 6.2.1.31 void dsp_chmul ([stpDCplx](#) *spDest*, double *dSrc*, long *lCount*)
- 6.2.1.32 void dsp_cmulf ([stpSCplx](#) *spDest*, [stSCplx](#) *sSrc*, long *lCount*)
- 6.2.1.33 void dsp_cmul ([stpDCplx](#) *spDest*, [stDCplx](#) *sSrc*, long *lCount*)
- 6.2.1.34 void dsp_mulf_nip (float * *fpDest*, const float * *fpSrc1*, float *fSrc2*, long *lCount*)
- 6.2.1.35 void dsp_mul_nip (double * *dpDest*, const double * *dpSrc1*, double *dSrc2*, long *lCount*)
- 6.2.1.36 void dsp_mul2f (float * *fpDest*, const float * *fpSrc*, long *lCount*)
- 6.2.1.37 void dsp_mul2 (double * *dpDest*, const double * *dpSrc*, long *lCount*)
- 6.2.1.38 void dsp_chmul2f ([stpSCplx](#) *spDest*, const float * *fpSrc*, long *lCount*)
- 6.2.1.39 void dsp_chmul2 ([stpDCplx](#) *spDest*, const double * *dpSrc*, long *lCount*)
- 6.2.1.40 void dsp_cmulf2 ([stpSCplx](#) *spDest*, const [stpSCplx](#) *spSrc*, long *lCount*)
- 6.2.1.41 void dsp_cmul2 ([stpDCplx](#) *spDest*, const [stpDCplx](#) *spSrc*, long *lCount*)
- 6.2.1.42 void dsp_mul3f (float * *fpDest*, const float * *fpSrc1*, const float * *fpSrc2*, long *lCount*)
- 6.2.1.43 void dsp_mul3 (double * *dpDest*, const double * *dpSrc1*, const double * *dpSrc2*, long *lCount*)
- 6.2.1.44 void dsp_cmulf3 ([stpSCplx](#) *spDest*, const [stpSCplx](#) *spSrc1*, const [stpSCplx](#) *spSrc2*, long *lCount*)
- 6.2.1.45 void dsp_cmul3 ([stpDCplx](#) *spDest*, const [stpDCplx](#) *spSrc1*, const [stpDCplx](#) *spSrc2*, long *lCount*)
- 6.2.1.46 void dsp_ccmulf ([stpSCplx](#) *spDest*, const [stpSCplx](#) *spSrc*, long *lCount*)
- 6.2.1.47 void dsp_ccmul ([stpDCplx](#) *spDest*, const [stpDCplx](#) *spSrc*, long *lCount*)
- 6.2.1.48 void dsp_ccmulf_nip ([stpSCplx](#) *spDest*, const [stpSCplx](#) *spSrc1*, const [stpSCplx](#) *spSrc2*, long *lCount*)
- 6.2.1.49 void dsp_ccmul_nip ([stpDCplx](#) *spDest*, const [stpDCplx](#) *spSrc1*, const [stpDCplx](#) *spSrc2*, long *lCount*)
- 6.2.1.50 void dsp_divf (float * *fpDest*, float *fSrc*, long *lCount*)

Generated on Tue Mar 2 19:56:37 2004 for libDSP by Doxygen

- 6.2.1.51 void dsp_div (double * *dpDest*, double *dSrc*, long *lCount*)
- 6.2.1.52 void dsp_cdivf ([stpSCplx](#) *spDest*, [stSCplx](#) *sSrc*, long *lCount*)
- 6.2.1.53 void dsp_cdiv ([stpDCplx](#) *spDest*, [stDCplx](#) *sSrc*, long *lCount*)
- 6.2.1.54 void dsp_div2f (float * *fpDest*, const float * *fpSrc*, long *lCount*)
- 6.2.1.55 void dsp_div2 (double * *dpDest*, const double * *dpSrc*, long *lCount*)
- 6.2.1.56 void dsp_cdiv2f ([stpSCplx](#) *spDest*, const [stpSCplx](#) *spSrc*, long *lCount*)
- 6.2.1.57 void dsp_cdiv2 ([stpDCplx](#) *spDest*, const [stpDCplx](#) *spSrc*, long *lCount*)
- 6.2.1.58 void dsp_div3f (float * *fpDest*, const float * *fpSrc1*, const float * *fpSrc2*, long *lCount*)
- 6.2.1.59 void dsp_div3 (double * *dpDest*, const double * *dpSrc1*, const double * *dpSrc2*, long *lCount*)
- 6.2.1.60 void dsp_cdiv3f ([stpSCplx](#) *spDest*, const [stpSCplx](#) *spSrc1*, const [stpSCplx](#) *spSrc2*, long *lCount*)
- 6.2.1.61 void dsp_cdiv3 ([stpDCplx](#) *spDest*, const [stpDCplx](#) *spSrc1*, const [stpDCplx](#) *spSrc2*, long *lCount*)
- 6.2.1.62 void dsp_div1xf (float * *fpVect*, long *lCount*)

[clDSPOp::Div1x](#)

- 6.2.1.63 void dsp_div1x (double * *dpVect*, long *lCount*)
- 6.2.1.64 void dsp_div1xf_nip (float * *fpDest*, const float * *fpSrc*, long *lCount*)
- 6.2.1.65 void dsp_div1x_nip (double * *dpDest*, const double * *dpSrc*, long *lCount*)
- 6.2.1.66 void dsp_muladdf (float * *fpVect*, float *fMul*, float *fAdd*, long *lCount*)

[clDSPOp::MulAdd](#)

6.2.1.67 void dsp_muladd (double * *dpVect*, double *dMul*, double *dAdd*, long *lCount*)

6.2.1.68 void dsp_muladdf_nip (float * *fpDest*, const float * *fpSrc*, float *fMul*, float *fAdd*, long *lCount*)

6.2.1.69 void dsp_muladd_nip (double * *dpDest*, const double * *dpSrc*, double *dMul*, double *dAdd*, long *lCount*)

6.2.1.70 void dsp_absf (float * *fpVect*, long *lCount*)

[clDSPOp::Abs](#)

6.2.1.71 void dsp_abs (double * *dpVect*, long *lCount*)

6.2.1.72 void dsp_absf_nip (float * *fpDest*, const float * *fpSrc*, long *lCount*)

6.2.1.73 void dsp_abs_nip (double * *dpDest*, const double * *dpSrc*, long *lCount*)

6.2.1.74 void dsp_sqrtf (float * *fpVect*, long *lCount*)

[clDSPOp::Sqrt](#)

6.2.1.75 void dsp_sqrt (double * *dpVect*, long *lCount*)

6.2.1.76 void dsp_sqrtf_nip (float * *fpDest*, const float * *fpSrc*, long *lCount*)

6.2.1.77 void dsp_sqrt_nip (double * *dpDest*, const double * *dpSrc*, long *lCount*)

6.2.1.78 void dsp_zerof (float * *fpVect*, long *lCount*)

[clDSPOp::Zero](#)

6.2.1.79 void dsp_zero (double * *dpVect*, long *lCount*)

6.2.1.80 void dsp_czerof ([stpSCplx](#) *spVect*, long *lCount*)

6.2.1.81 void dsp_czero ([stpDCplx](#) *spVect*, long *lCount*)

6.2.1.82 void dsp_setf (float * *fpDest*, float *fSrc*, long *lCount*)

[clDSPOp::Set](#)

- 6.2.1.83 void dsp_set (double * *dpDest*, double *dSrc*, long *lCount*)
- 6.2.1.84 void dsp_csetf ([stpSCplx](#) *spDest*, [stSCplx](#) *sSrc*, long *lCount*)
- 6.2.1.85 void dsp_cset ([stpDCplx](#) *spDest*, [stDCplx](#) *sSrc*, long *lCount*)
- 6.2.1.86 void dsp_set2f (float * *fpDest*, float *fSrc*, long *lStart*, long *lCount*, long *lLength*)
- 6.2.1.87 void dsp_set2 (double * *dpDest*, double *dSrc*, long *lStart*, long *lCount*, long *lLength*)
- 6.2.1.88 void dsp_cset2f ([stpSCplx](#) *spDest*, [stSCplx](#) *sSrc*, long *lStart*, long *lCount*, long *lLength*)
- 6.2.1.89 void dsp_cset2 ([stpDCplx](#) *spDest*, [stDCplx](#) *sSrc*, long *lStart*, long *lCount*, long *lLength*)
- 6.2.1.90 void dsp_clipf (float * *fpVect*, float *fValue*, long *lCount*)

[clDSPOp::Clip](#)

- 6.2.1.91 void dsp_clip (double * *dpVect*, double *dValue*, long *lCount*)
- 6.2.1.92 void dsp_clipf_nip (float * *fpDest*, const float * *fpSrc*, float *fValue*, long *lCount*)
- 6.2.1.93 void dsp_clip_nip (double * *dpDest*, const double * *dpSrc*, double *dValue*, long *lCount*)
- 6.2.1.94 void dsp_clip2f (float * *fpVect*, float *fMin*, float *fMax*, long *lCount*)
- 6.2.1.95 void dsp_clip2 (double * *dpVect*, double *dMin*, double *dMax*, long *lCount*)
- 6.2.1.96 void dsp_clip2f_nip (float * *fpDest*, const float * *fpSrc*, float *fMin*, float *fMax*, long *lCount*)
- 6.2.1.97 void dsp_clip2_nip (double * *dpDest*, const double * *dpSrc*, double *dMin*, double *dMax*, long *lCount*)
- 6.2.1.98 void dsp_clipzerof (float * *fpVect*, long *lCount*)

[clDSPOp::ClipZero](#)

6.2.1.99 void dsp_clipzero (double * *dpVect*, long *lCount*)

6.2.1.100 void dsp_clipzerof_nip (float * *fpDest*, const float * *fpSrc*, long *lCount*)

6.2.1.101 void dsp_clipzero_nip (double * *dpDest*, const double * *dpSrc*, long *lCount*)

6.2.1.102 void dsp_copyf (float * *fpDest*, const float * *fpSrc*, long *lCount*)

[cIDSPOp::Copy](#)

6.2.1.103 void dsp_copy (double * *dpDest*, const double * *dpSrc*, long *lCount*)

6.2.1.104 void dsp_ccopyf ([stpSCplx](#) *spDest*, const [stpSCplx](#) *spSrc*, long *lCount*)

6.2.1.105 void dsp_ccopy ([stpDCplx](#) *spDest*, const [stpDCplx](#) *spSrc*, long *lCount*)

6.2.1.106 float dsp_convolvef (const float * *fpSrc1*, const float * *fpSrc2*, long *lCount*)

[cIDSPOp::Convolve](#)

6.2.1.107 double dsp_convolve (const double * *dpSrc1*, const double * *dpSrc2*, long *lCount*)

6.2.1.108 void dsp_convolve2f (float * *fpDest*, const float * *fpSrc1*, const float * *fpSrc2*, long *lCount*)

6.2.1.109 void dsp_convolve2 (double * *dpDest*, const double * *dpSrc1*, const double * *dpSrc2*, long *lCount*)

6.2.1.110 float dsp_correlatef (const float * *fpSrc1*, const float * *fpSrc2*, long *lCount*)

[cIDSPOp::Correlate](#)

- 6.2.1.111** `double dsp_correlate (const double * dpSrc1, const double * dpSrc2, long lCount)`
- 6.2.1.112** `void dsp_correlate2f (float * fpDest, const float * fpSrc1, const float * fpSrc2, long lCount)`
- 6.2.1.113** `void dsp_correlate2 (double * dpDest, const double * dpSrc1, const double * dpSrc2, long lCount)`
- 6.2.1.114** `float dsp_autocorrf (const float * fpSrc, long lCount)`

[clDSPOp::AutoCorrelate](#)

- 6.2.1.115** `double dsp_autocorr (const double * dpSrc, long lCount)`
- 6.2.1.116** `void dsp_autocorr2f (float * fpDest, const float * fpSrc, long lCount)`
- 6.2.1.117** `void dsp_autocorr2 (double * dpDest, const double * dpSrc, long lCount)`
- 6.2.1.118** `float dsp_dotproductf (const float * fpSrc1, const float * fpSrc2, long lCount)`

[clDSPOp::DotProduct](#)

- 6.2.1.119** `double dsp_dotproduct (const double * dpSrc1, const double * dpSrc2, long lCount)`
- 6.2.1.120** `void dsp_minmaxf (float * fpMin, float * fpMax, const float * fpSrc, long lCount)`

[clDSPOp::MinMax](#)

- 6.2.1.121** `void dsp_minmax (double * dpMin, double * dpMax, const double * dpSrc, long lCount)`
- 6.2.1.122** `float dsp_meanf (const float * fpSrc, long lCount)`

[clDSPOp::Mean](#)

6.2.1.123 `double dsp_mean (const double * dpSrc, long lCount)`

6.2.1.124 `float dsp_medianf (const float * fpSrc, long lCount)`

6.2.1.125 `double dsp_median (const double * dpSrc, long lCount)`

6.2.1.126 `void dsp_negatef (float * fpVect, long lCount)`

[clDSPOp::Negate](#)

6.2.1.127 `void dsp_negate (double * dpVect, long lCount)`

6.2.1.128 `void dsp_negatef_nip (float * fpDest, const float * fpSrc, long lCount)`

6.2.1.129 `void dsp_negate_nip (double * dpDest, const double * dpSrc, long lCount)`

6.2.1.130 `void dsp_normalizef (float * fpVect, long lCount)`

[clDSPOp::Normalize](#)

6.2.1.131 `void dsp_normalize (double * dpVect, long lCount)`

6.2.1.132 `void dsp_normalizef_nip (float * fpDest, const float * fpSrc, long lCount)`

6.2.1.133 `void dsp_normalize_nip (double * dpDest, const double * dpSrc, long lCount)`

6.2.1.134 `float dsp_productf (const float * fpSrc, long lCount)`

[clDSPOp::Product](#)

6.2.1.135 `double dsp_product (const double * dpSrc, long lCount)`

6.2.1.136 `void dsp_reversef (float * fpVect, long lCount)`

[clDSPOp::Reverse](#)

6.2.1.137 void dsp_reverse (double * *dpVect*, long *lCount*)

6.2.1.138 void dsp_reversef_nip (float * *fpDest*, const float * *fpSrc*, long *lCount*)

6.2.1.139 void dsp_reverse_nip (double * *dpDest*, const double * *dpSrc*, long *lCount*)

6.2.1.140 void dsp_scalef (float * *fpVect*, long *lCount*)

[cldSPOp::Scale](#)

6.2.1.141 void dsp_scale (double * *dpVect*, long *lCount*)

6.2.1.142 void dsp_scalef_nip (float * *fpDest*, const float * *fpSrc*, long *lCount*)

6.2.1.143 void dsp_scale_nip (double * *dpDest*, const double * *dpSrc*, long *lCount*)

6.2.1.144 void dsp_scale01f (float * *fpVect*, long *lCount*)

[cldSPOp::Scale01](#)

6.2.1.145 void dsp_scale01 (double * *dpVect*, long *lCount*)

6.2.1.146 void dsp_scale01f_nip (float * *fpDest*, const float * *fpSrc*, long *lCount*)

6.2.1.147 void dsp_scale01_nip (double * *dpDest*, const double * *dpSrc*, long *lCount*)

6.2.1.148 void dsp_sortf (float * *fpVect*, long *lCount*)

[cldSPOp::Sort](#)

6.2.1.149 void dsp_sort (double * *dpVect*, long *lCount*)

6.2.1.150 void dsp_sortl (long * *lpVect*, long *lCount*)

6.2.1.151 void dsp_stddevf (float * *fpStdDev*, float * *fpMean*, const float * *fpSrc*, long *lCount*)

[cldSPOp::StdDev](#)

6.2.1.152 void dsp_stddev (double * *dpStdDev*, double * *dpMean*, const double * *dpSrc*, long *lCount*)

6.2.1.153 float dsp_sumf (const float * *fpSrc*, long *lCount*)

[clDSPOp::Sum](#)

6.2.1.154 double dsp_sum (const double * *dpSrc*, long *lCount*)

6.2.1.155 void dsp_squaref (float * *fpVect*, long *lCount*)

[clDSPOp::Square](#)

6.2.1.156 void dsp_square (double * *dpVect*, long *lCount*)

6.2.1.157 void dsp_squaref_nip (float * *fpDest*, const float * *fpSrc*, long *lCount*)

6.2.1.158 void dsp_square_nip (double * *dpDest*, const double * *dpSrc*, long *lCount*)

6.2.1.159 void dsp_convertu8f (float * *fpDest*, const unsigned char * *ucpSrc*, long *lCount*)

[clDSPOp::Convert](#)

- 6.2.1.160 void dsp_convertu8 (double * *dpDest*, const unsigned char * *ucpSrc*, long *lCount*)
- 6.2.1.161 void dsp_converts16f (float * *fpDest*, const signed short * *ipSrc*, long *lCount*, int *i12bit*)
- 6.2.1.162 void dsp_converts16 (double * *dpDest*, const signed short * *ipSrc*, long *lCount*, int *i12bit*)
- 6.2.1.163 void dsp_converts32f (float * *fpDest*, const signed int * *ipSrc*, long *lCount*, int *i24bit*)
- 6.2.1.164 void dsp_converts32 (double * *dpDest*, const signed int * *ipSrc*, long *lCount*, int *i24bit*)
- 6.2.1.165 void dsp_converttd64f (float * *fpDest*, const double * *dpSrc*, long *lCount*)
- 6.2.1.166 void dsp_convertf32 (double * *dpDest*, const float * *fpSrc*, long *lCount*)
- 6.2.1.167 void dsp_convertf32c (unsigned char * *ucpDest*, const float * *fpSrc*, long *lCount*)
- 6.2.1.168 void dsp_converttd64c (unsigned char * *ucpDest*, const double * *dpSrc*, long *lCount*)
- 6.2.1.169 void dsp_convertf32s (signed short * *ipDest*, const float * *fpSrc*, long *lCount*, int *i12bit*)
- 6.2.1.170 void dsp_converttd64s (signed short * *ipDest*, const double * *dpSrc*, long *lCount*, int *i12bit*)
- 6.2.1.171 void dsp_convertf32i (signed int * *ipDest*, const float * *fpSrc*, long *lCount*, int *i24bit*)
- 6.2.1.172 void dsp_converttd64i (signed int * *ipDest*, const double * *dpSrc*, long *lCount*, int *i24bit*)
- 6.2.1.173 void dsp_cart2polarf (float * *fpMagn*, float * *fpPhase*, const float * *fpReal*, const float * *fpImag*, long *lCount*)

[cdDSPOp::CartToPolar](#)

- 6.2.1.174 void dsp_cart2polar (double * *dpMagn*, double * *dpPhase*, const double * *dpReal*, const double * *dpImag*, long *lCount*)
- 6.2.1.175 void dsp_cart2polar2f (float * *fpMagn*, float * *fpPhase*, const [stpSCplx](#) *spCart*, long *lCount*)
- 6.2.1.176 void dsp_cart2polar2 (double * *dpMagn*, double * *dpPhase*, const [stpDCplx](#) *spCart*, long *lCount*)
- 6.2.1.177 void dsp_cart2polar3f ([stpSPolar](#) *spPolar*, const [stpSCplx](#) *spCart*, long *lCount*)
- 6.2.1.178 void dsp_cart2polar3 ([stpDPolar](#) *spPolar*, const [stpDCplx](#) *spCart*, long *lCount*)
- 6.2.1.179 void dsp_cart2polar4f ([utpSCoord](#) *upCoord*, long *lCount*)
- 6.2.1.180 void dsp_cart2polar4 ([utpDCoord](#) *upCoord*, long *lCount*)
- 6.2.1.181 void dsp_polar2cartf (float * *fpReal*, float * *fpImag*, const float * *fpMagn*, const float * *fpPhase*, long *lCount*)

[clDSPOp::PolarToCart](#)

- 6.2.1.182 void dsp_polar2cart (double * *dpReal*, double * *dpImag*, const double * *dpMagn*, const double * *dpPhase*, long *lCount*)
- 6.2.1.183 void dsp_polar2cart2f ([stpSCplx](#) *spCart*, const float * *fpMagn*, const float * *fpPhase*, long *lCount*)
- 6.2.1.184 void dsp_polar2cart2 ([stpDCplx](#) *spCart*, const double * *dpMagn*, const double * *dpPhase*, long *lCount*)
- 6.2.1.185 void dsp_polar2cart3f ([stpSCplx](#) *spCart*, const [stpSPolar](#) *spPolar*, long *lCount*)
- 6.2.1.186 void dsp_polar2cart3 ([stpDCplx](#) *spCart*, const [stpDPolar](#) *spPolar*, long *lCount*)
- 6.2.1.187 void dsp_polar2cart4f ([utpSCoord](#) *upCoord*, long *lCount*)
- 6.2.1.188 void dsp_polar2cart4 ([utpDCoord](#) *upCoord*, long *lCount*)
- 6.2.1.189 float dsp_crosscorrf (const float * *fpSrc1*, const float * *fpSrc2*, long *lCount*)

[clDSPOp::CrossCorr](#)

6.2.1.191 float dsp_crosscorr2f (const float * *fpSrc1*, const float * *fpSrc2*, long *lDelay*, long *lCount*)

6.2.1.195 float dsp_energyf (const float * *fpSrc*, long *lCount*)

6.2.1.199 void dsp_powerf (float *fpDest, const stpSCplx spSrc, long lCount)

clDSPOp::Phase

6.2.1.202 void dsp_phase (double * *dpDest*, const [stpDCplx](#) *spSrc*, long *lCount*)

6.2.1.203 void dsp_powerphasef (float * *fpPower*, float * *fpPhase*, const [stpSCplx](#) *spSrc*, long *lCount*)

[cIDSPOp::PowerPhase](#)

6.2.1.204 void dsp_powerphase (double * *dpPower*, double * *dpPhase*, const [stpDCplx](#) *spSrc*, long *lCount*)

6.2.1.205 void dsp_decimatef (float * *fpDest*, const float * *fpSrc*, long *lFactor*, long *lCount*)

[cIDSPOp::Decimate](#)

6.2.1.206 void dsp_decimate (double * *dpDest*, const double * *dpSrc*, long *lFactor*, long *lCount*)

6.2.1.207 void dsp_decimateavgf (float * *fpDest*, const float * *fpSrc*, long *lFactor*, long *lCount*)

[cIDSPOp::DecimateAvg](#)

6.2.1.208 void dsp_decimateavg (double * *dpDest*, const double * *dpSrc*, long *lFactor*, long *lCount*)

6.2.1.209 void dsp_interpolatef (float * *fpDest*, const float * *fpSrc*, long *lFactor*, long *lCount*)

[cIDSPOp::Interpolate](#)

6.2.1.210 void dsp_interpolate (double * *dpDest*, const double * *dpSrc*, long *lFactor*, long *lCount*)

6.2.1.211 void dsp_resamplef (float * *fpDest*, long *lDestCount*, const float * *fpSrc*, long *lSrcCount*)

[cIDSPOp::Resample](#)

6.2.1.212 void dsp_resample (double * *dpDest*, long *lDestCount*, const double * *dpSrc*, long *lSrcCount*)

6.2.1.213 void dsp_resampleavgf (float * *fpDest*, long *lDestCount*, const float * *fpSrc*, long *lSrcCount*)

[cIDSPOp::ResampleAvg](#)

6.2.1.214 void dsp_resampleavg (double * *dpDest*, long *lDestCount*, const double * *dpSrc*, long *lSrcCount*)

6.2.1.215 void dsp_interpolateavgf (float * *fpDest*, const float * *fpSrc*, long *lFactor*, long *lCount*)

[cIDSPOp::InterpolateAvg](#)

6.2.1.216 void dsp_interpolateavg (double * *dpDest*, const double * *dpSrc*, long *lFactor*, long *lCount*)

6.2.1.217 float dsp_rmsf (const float * *fpSrc*, long *lCount*)

[cIDSPOp::RMS](#)

6.2.1.218 double dsp_rms (const double * *dpSrc*, long *lCount*)

6.2.1.219 float dsp_variancef (float * *fpVariance*, float * *fpMean*, const float * *fpSrc*, long *lCount*)

[cIDSPOp::Variance](#)

6.2.1.220 double dsp_variance (double * *dpVariance*, double * *dpMean*, const double * *dpSrc*, long *lCount*)

6.2.1.221 float dsp_peaklevelf (const float * *fpSrc*, long *lCount*)

[cIDSPOp::PeakLevel](#)

6.2.1.222 double dsp_peaklevel (const double * *dpSrc*, long *lCount*)

6.2.1.223 void dsp_mixf (float * *fpDest*, const float * *fpSrc*, long *lCount*)

[cIDSPOp::Mix](#)

- 6.2.1.224** void `dsp_mix` (`double * dpDest`, `const double * dpSrc`, `long lCount`)
- 6.2.1.225** void `dsp_mix2f` (`float * fpDest`, `const float * fpSrc1`, `const float * fpSrc2`, `long lCount`)
- 6.2.1.226** void `dsp_mix2` (`double * dpDest`, `const double * dpSrc1`, `const double * dpSrc2`, `long lCount`)
- 6.2.1.227** void `dsp_mixnf` (`float * fpDest`, `const float * fpSrc`, `long lChCount`, `long lCount`)
- 6.2.1.228** void `dsp_mixn` (`double * dpDest`, `const double * dpSrc`, `long lChCount`, `long lCount`)
- 6.2.1.229** void `dsp_extractf` (`float * fpDest`, `const float * fpSrc`, `long lChannel`, `long lChCount`, `long lCount`)

[clDSPOp::Extract](#)

- 6.2.1.230** void `dsp_extract` (`double * dpDest`, `const double * dpSrc`, `long lChannel`, `long lChCount`, `long lCount`)
- 6.2.1.231** void `dsp_packf` (`float * fpDest`, `const float * fpSrc`, `long lChannel`, `long lChCount`, `long lCount`)

[clDSPOp::Pack](#)

- 6.2.1.232** void `dsp_pack` (`double * dpDest`, `const double * dpSrc`, `long lChannel`, `long lChCount`, `long lCount`)
- 6.2.1.233** void `dsp_fftw_convertf2cf` ([stpSCplx](#) `spDest`, `const float * fpSrc`, `long lCount`)

[clDSPOp::FFTWConvert](#)

- 6.2.1.234 void `dsp_fftw_convertf2cd` ([stpDCplx](#) *spDest*, const float * *fpSrc*, long *lCount*)
- 6.2.1.235 void `dsp_fftw_convertd2cf` ([stpSCplx](#) *spDest*, const double * *dpSrc*, long *lCount*)
- 6.2.1.236 void `dsp_fftw_convertd2cd` ([stpDCplx](#) *spDest*, const double * *dpSrc*, long *lCount*)
- 6.2.1.237 void `dsp_fftw_convertcf2f` (float * *fpDest*, const [stpSCplx](#) *spSrc*, long *lCount*)
- 6.2.1.238 void `dsp_fftw_convertcd2f` (float * *fpDest*, const [stpDCplx](#) *spSrc*, long *lCount*)
- 6.2.1.239 void `dsp_fftw_convertcf2d` (double * *dpDest*, const [stpSCplx](#) *spSrc*, long *lCount*)
- 6.2.1.240 void `dsp_fftw_convertcd2d` (double * *dpDest*, const [stpDCplx](#) *spSrc*, long *lCount*)

6.2.1.241 [dsp_t](#) `dsp_new` (void)

Creates new instance of DSP object.

Returns:

DSP object instance

6.2.1.242 void `dsp_delete` ([dsp_t](#))

Deletes DSP object instance.

Parameters:

dspInst DSP object instance

6.2.1.243 void `dsp_win_bartlett` ([dsp_t](#) *dspInst*, float * *fpDest*, long *lCount*)

[clDSPOp::WinBartlett](#)

6.2.1.244 void `dsp_win_bartlett` ([dsp_t](#) *dspInst*, double * *dpDest*, long *lCount*)

6.2.1.245 void `dsp_win_blackmanf` ([dsp_t](#) *dspInst*, float * *fpDest*, long *lCount*, float *fAlpha*)

[clDSPOp::WinBlackman](#)

6.2.1.246 void dsp_win_blackman ([dsp_t](#) *dspInst*, double * *dpDest*, long *lCount*, double *dAlpha*)

6.2.1.247 void dsp_win_blackman_harrisf ([dsp_t](#) *dspInst*, float * *fpDest*, long *lCount*)

[clDSPOp::WinBlackmanHarris](#)

6.2.1.248 void dsp_win_blackman_harris ([dsp_t](#) *dspInst*, double * *dpDest*, long *lCount*)

6.2.1.249 void dsp_win_cos_taperedf ([dsp_t](#) *dspInst*, float * *fpDest*, long *lCount*)

[clDSPOp::WinCosTapered](#)

6.2.1.250 void dsp_win_cos_tapered ([dsp_t](#) *dspInst*, double * *dpDest*, long *lCount*)

6.2.1.251 void dsp_win_exact_blackmanf ([dsp_t](#) *dspInst*, float * *fpDest*, long *lCount*)

[clDSPOp::WinExactBlackman](#)

6.2.1.252 void dsp_win_exact_blackman ([dsp_t](#) *dspInst*, double * *dpDest*, long *lCount*)

6.2.1.253 void dsp_win_flat_topf ([dsp_t](#) *dspInst*, float * *fpDest*, long *lCount*)

[clDSPOp::WinFlatTop](#)

6.2.1.254 void dsp_win_flat_top ([dsp_t](#) *dspInst*, double * *dpDest*, long *lCount*)

6.2.1.255 void dsp_win_generic_cosf ([dsp_t](#) *dspInst*, float * *fpDest*, long *lCount*, const float * *fpCoeff*, long *lCoeffCount*)

[clDSPOp::WinGenericCos](#)

6.2.1.256 void dsp_win_generic_cos ([dsp_t](#) *dspInst*, double * *dpDest*, long *lCount*, const double * *dpCoeff*, long *lCoeffCount*)

6.2.1.257 void dsp_win_hammingf ([dsp_t](#) *dspInst*, float * *fpDest*, long *lCount*)

[clDSPOp::WinHamming](#)

6.2.1.258 void dsp_win_hamming (**dsp_t** *dspInst*, double * *dpDest*, long *lCount*)

6.2.1.259 void dsp_win_hanningf (**dsp_t** *dspInst*, float * *fpDest*, long *lCount*)

[cldSPOp::WinHanning](#)

6.2.1.260 void dsp_win_hanning (**dsp_t** *dspInst*, double * *dpDest*, long *lCount*)

6.2.1.261 void dsp_win_kaiserf (**dsp_t** *dspInst*, float * *fpDest*, float *fBeta*, long *lCount*)

[cldSPOp::WinKaiser](#)

6.2.1.262 void dsp_win_kaiser (**dsp_t** *dspInst*, double * *dpDest*, double *dBeta*, long *lCount*)

6.2.1.263 void dsp_win_kaiser_besself (**dsp_t** *dspInst*, float * *fpDest*, float *fAlpha*, long *lCount*)

[cldSPOp::WinKaiserBessel](#)

6.2.1.264 void dsp_win_kaiser_bessel (**dsp_t** *dspInst*, double * *dpDest*, double *dAlpha*, long *lCount*)

6.2.1.265 void dsp_win_tukeyf (**dsp_t** *dspInst*, float * *fpDest*, long *lCount*)

[cldSPOp::WinTukey](#)

6.2.1.266 void dsp_win_tukey (**dsp_t** *dspInst*, double * *dpDest*, long *lCount*)

6.2.1.267 void dsp_win_dolph_chebyshevf (**dsp_t** *dspInst*, float * *fpDest*, float *fGamma*, long *lCount*)

[cldSPOp::WinDolphChebyshev](#)

6.2.1.268 void dsp_win_dolph_chebyshev (**dsp_t** *dspInst*, double * *dpDest*, double *dGamma*, long *lCount*)

6.2.1.269 long dsp_rebufferf (**dsp_t** *dspInst*, float * *fpDest*, const float * *fpSrc*, long *lDestCount*, long *lSrcCount*)

[cldSPOp::ReBuffer](#)

6.2.1.270 long dsp_rebuffer ([dsp_t](#) *dspInst*, double * *dpDest*, const double * *dpSrc*, long *lDestCount*, long *lSrcCount*)

6.2.1.271 float dsp_deg2radf ([dsp_t](#) *dspInst*, float *fSource*)

[clDSPOp::DegToRad](#)

6.2.1.272 double dsp_deg2rad ([dsp_t](#) *dspInst*, double *dSource*)

6.2.1.273 float dsp_rad2degf ([dsp_t](#) *dspInst*, float *fSource*)

[clDSPOp::RadToDeg](#)

6.2.1.274 double dsp_rad2deg ([dsp_t](#) *dspInst*, double *dSource*)

6.2.1.275 void dsp_fir_allocatf ([dsp_t](#) *dspInst*, const float * *fpCoeff*, long *lCount*)

[clDSPOp::FIRAllocate](#)

6.2.1.276 void dsp_fir_allocate ([dsp_t](#) *dspInst*, const double * *dpCoeff*, long *lCount*)

6.2.1.277 void dsp_fir_free ([dsp_t](#) *dspInst*)

[clDSPOp::FIRFree](#)

6.2.1.278 void dsp_fir_filterf ([dsp_t](#) *dspInst*, float * *fpVect*, long *lCount*)

[clDSPOp::FIRFilter](#)

6.2.1.279 void dsp_fir_filter ([dsp_t](#) *dspInst*, double * *dpVect*, long *lCount*)

6.2.1.280 void dsp_fir_filterf_nip ([dsp_t](#) *dspInst*, float * *fpDest*, const float * *fpSrc*, long *lCount*)

6.2.1.281 void dsp_fir_filter_nip ([dsp_t](#) *dspInst*, double * *dpDest*, const double * *dpSrc*, long *lCount*)

6.2.1.282 void dsp_fir_filterf_fst ([dsp_t](#) *dspInst*, float * *fpDest*, float * *fpSrc*, long *lCount*)

[clDSPOp::FIRFilterF](#)

6.2.1.283 void dsp_fir_filter_fst ([dsp_t](#) *dspInst*, double * *dpDest*, double * *dpSrc*, long *lCount*)

6.2.1.284 void dsp_iir_initf ([dsp_t](#) *dspInst*, const float * *fpCoeffs*)

[clDSPOp::IIRInitialize](#)

6.2.1.285 void dsp_iir_init ([dsp_t](#) *dspInst*, const double * *dpCoeffs*)

6.2.1.286 void dsp_iir_filterf ([dsp_t](#) *dspInst*, float * *fpVect*, long *lCount*)

[clDSPOp::IIRFilter](#)

6.2.1.287 void dsp_iir_filter ([dsp_t](#) *dspInst*, double * *dpVect*, long *lCount*)

6.2.1.288 void dsp_iir_filterf_nip ([dsp_t](#) *dspInst*, float * *fpDest*, const float * *fpSrc*, long *lCount*)

6.2.1.289 void dsp_iir_filter_nip ([dsp_t](#) *dspInst*, double * *dpDest*, const double * *dpSrc*, long *lCount*)

6.2.1.290 void dsp_iir_clear ([dsp_t](#) *dspInst*)

[clDSPOp::IIRCLEAR](#)

6.2.1.291 void dsp_fft_init ([dsp_t](#) *dspInst*, long *lSize*, int *iReal*)

[clDSPOp::FFTInitialize](#)

6.2.1.292 void dsp_fft_uninit ([dsp_t](#) *dspInst*)

[clDSPOp::FFTUninitialize](#)

6.2.1.293 void dsp_fftf ([dsp_t](#) *dspInst*, [stpSCplx](#) *spDest*, float * *fpSrc*)

[clDSPOp::FFTi](#)

6.2.1.294 void dsp_fft ([dsp_t](#) *dspInst*, [stpDCplx](#) *spDest*, double * *dpSrc*)

6.2.1.295 void dsp_fftf_nip ([dsp_t](#) *dspInst*, [stpSCplx](#) *spDest*, const float * *fpSrc*)

[clDSPOp::FFTTo](#)

- 6.2.1.296 void `dsp_fft_nip` (`dsp_t dspInst`, `stpDCplx spDest`, const double * `dpSrc`)
- 6.2.1.297 void `dsp_cfft_nip` (`dsp_t dspInst`, `stpSCplx spDest`, const `stpSCplx spSrc`)
- 6.2.1.298 void `dsp_cfft_nip` (`dsp_t dspInst`, `stpDCplx spDest`, const `stpDCplx spSrc`)
- 6.2.1.299 void `dsp_ifft_nip` (`dsp_t dspInst`, float * `fpDest`, const `stpSCplx spSrc`)

`clDSPOp::IFFTo`

- 6.2.1.300 void `dsp_ifft_nip` (`dsp_t dspInst`, double * `dpDest`, const `stpDCplx spSrc`)
- 6.2.1.301 void `dsp_cifft_nip` (`dsp_t dspInst`, `stpSCplx spDest`, const `stpSCplx spSrc`)
- 6.2.1.302 void `dsp_cifft_nip` (`dsp_t dspInst`, `stpDCplx spDest`, const `stpDCplx spSrc`)
- 6.2.1.303 `dsp_iircas_t dsp_iir_cas_new` (void)

Creates new instance of cascaded IIR object.

Returns:

Cascade IIR instance

- 6.2.1.304 void `dsp_iir_cas_delete` (`dsp_iircas_t`)

Deletes cascaded IIR object instance.

Parameters:

`dspiircasInst` Cascaded IIR instance

- 6.2.1.305 int `dsp_iir_cas_initf` (`dsp_iircas_t dspiircasInst`, const float `fpCoeffs`[[5], long `lStageCount`)

`clIIRCascade::Initialize`

- 6.2.1.306 `int dsp_iir_cas_init (dsp_iircas_t dspiircasInst, const double dpCoeffs[][5], long lStageCount)`
- 6.2.1.307 `void dsp_iir_cas_uninit (dsp_iircas_t dspiircasInst)`
- 6.2.1.308 `void dsp_iir_cas_processf (dsp_iircas_t dspiircasInst, float * fpVect, long lCount)`
- 6.2.1.309 `void dsp_iir_cas_process (dsp_iircas_t dspiircasInst, double * dpVect, long lCount)`
- 6.2.1.310 `void dsp_iir_cas_processf_nip (dsp_iircas_t dspiircasInst, float * fpDest, const float * fpSrc, long lCount)`
- 6.2.1.311 `void dsp_iir_cas_process_nip (dsp_iircas_t dspiircasInst, double * dpDest, const double * dpSrc, long lCount)`
- 6.2.1.312 `void dsp_iir_cas_clear (dsp_iircas_t dspiircasInst)`
- 6.2.1.313 `dsp_decfft_t dsp_dec_fft_new (void)`

Creates new instance of FFT decimator object.

Returns:

FFT decimator instance

- 6.2.1.314 `void dsp_dec_fft_delete (dsp_decfft_t)`

Deletes FFT decimator object instance.

Parameters:

dspdecfftInst FFT decimator instance

- 6.2.1.315 `int dsp_dec_fft_initf (dsp_decfft_t dspdecfftInst, long lDecFact, long lFiltSize, int iHighPass)`

[clFFTDecimator::Initialize](#)

- 6.2.1.316 `int dsp_dec_fft_init (dsp_decfft_t dspdecfftInst, long lDecFact, long lFiltSize, int iHighPass)`

- 6.2.1.317 `void dsp_dec_fft_uninit (dsp_decfft_t dspdecfftInst)`

[clFFTDecimator::Uninitialize](#)

6.2.1.318 void dsp_dec_fft_putf ([dsp_deccfft_t](#) *dspdecfftInst*, const float * *fpSrcData*, long *lSrcCount*)

[clIFFTDecimator::Put](#)

6.2.1.319 void dsp_dec_fft_put ([dsp_deccfft_t](#) *dspdecfftInst*, const double * *dpSrcData*, long *lSrcCount*)

6.2.1.320 int dsp_dec_fft_getf ([dsp_deccfft_t](#) *dspdecfftInst*, float * *fpDestData*, long *lDestCount*)

[clIFFTDecimator::Get](#)

6.2.1.321 int dsp_dec_fft_get ([dsp_deccfft_t](#) *dspdecfftInst*, double * *dpDestData*, long *lDestCount*)

6.2.1.322 [dsp_decfir_t](#) dsp_dec_fir_new (void)

Creates new instance of FIR decimator object.

Returns:

FIR decimator instance

6.2.1.323 void dsp_dec_fir_delete ([dsp_decfir_t](#))

Deletes FIR decimator object instance.

Parameters:

dspdecfirInst FIR decimator instance

6.2.1.324 int dsp_dec_fir_initf ([dsp_decfir_t](#) *dspdecfirInst*, long *lDecFact*, int *iHighPass*)

[clFIRDecimator::Initialize](#)

6.2.1.325 int dsp_dec_fir_init ([dsp_decfir_t](#) *dspdecfirInst*, long *lDecFact*, int *iHighPass*)

6.2.1.326 void dsp_dec_fir_uninit ([dsp_decfir_t](#) *dspdecfirInst*)

[clFIRDecimator::Uninitialize](#)

6.2.1.327 void dsp_dec_fir_putf ([dsp_decfir_t](#) *dspdecfirInst*, const float *
fpSrcData, long *lSrcCount*)

[clFIRDecimator::Put](#)

6.2.1.328 void dsp_dec_fir_put ([dsp_decfir_t](#) *dspdecfirInst*, const double *
dpSrcData, long *lSrcCount*)

6.2.1.329 int dsp_dec_fir_getf ([dsp_decfir_t](#) *dspdecfirInst*, float **fpDestData*,
long *lDestCount*)

[clFIRDecimator::Get](#)

6.2.1.330 int dsp_dec_fir_get ([dsp_decfir_t](#) *dspdecfirInst*, double **dpDestData*,
long *lDestCount*)

6.2.1.331 [dsp_deciir_t](#) dsp_dec_iir_new (void)

Creates new instance of IIR decimator object.

Returns:

IIR decimator instance

6.2.1.332 void dsp_dec_iir_delete ([dsp_deciir_t](#))

Deletes IIR decimator object instance.

Parameters:

dspdeciirInst IIR decimator instance

6.2.1.333 int dsp_dec_iir_initf ([dsp_deciir_t](#) *dspdeciirInst*, long *lDecFact*, int
iHighPass)

[clIIRDecimator::Initialize](#)

6.2.1.334 int dsp_dec_iir_init ([dsp_deciir_t](#) *dspdeciirInst*, long *lDecFact*, int
iHighPass)

6.2.1.335 void dsp_dec_iir_uninit ([dsp_deciir_t](#) *dspdeciirInst*)

[clIIRDecimator::Uninitialize](#)

6.2.1.336 void dsp_dec_iir_putf ([dsp_deciir_t](#) *dspdeciirInst*, const float * *fpSrcData*, long *lSrcCount*)

[clIIRDecimator::Put](#)

6.2.1.337 void dsp_dec_iir_put ([dsp_deciir_t](#) *dspdeciirInst*, const double * *dpSrcData*, long *lSrcCount*)

6.2.1.338 int dsp_dec_iir_getf ([dsp_deciir_t](#) *dspdeciirInst*, float * *fpDestData*, long *lDestCount*)

[clIIRDecimator::Get](#)

6.2.1.339 int dsp_dec_iir_get ([dsp_deciir_t](#) *dspdeciirInst*, double * *dpDestData*, long *lDestCount*)

6.2.1.340 [dsp_decrec_t](#) dsp_dec_rec_new (void)

Creates new instance of recursive decimator object.

Returns:

Recursive decimator instance

6.2.1.341 void dsp_dec_rec_delete ([dsp_decrec_t](#))

Deletes recursive decimator object instance.

Parameters:

dspdecrecInst Recursive decimator instance

6.2.1.342 int dsp_dec_rec_initf ([dsp_decrec_t](#) *dspdecrecInst*, long *lDecFact*, long *lFiltSize*, float *fBandCenter*, int *iFilterType*)

[clRecDecimator::Initialize](#)

6.2.1.343 int dsp_dec_rec_init ([dsp_decrec_t](#) *dspdecrecInst*, long *lDecFact*, long *lFiltSize*, double *dBandCenter*, int *iFilterType*)

6.2.1.344 void dsp_dec_rec_uninit ([dsp_decrec_t](#) *dspdecrecInst*)

[clRecDecimator::Uninitialize](#)

6.2.1.345 void `dsp_dec_rec_putf` ([dsp_decrec_t](#) *dspdecrecInst*, const float * *fpSrcData*, long *lSrcCount*)

[clRecDecimator::Put](#)

6.2.1.346 void `dsp_dec_rec_put` ([dsp_decrec_t](#) *dspdecrecInst*, const double * *dpSrcData*, long *lSrcCount*)

6.2.1.347 int `dsp_dec_rec_getf` ([dsp_decrec_t](#) *dspdecrecInst*, float * *fpDestData*, long *lDestCount*)

[clRecDecimator::Get](#)

6.2.1.348 int `dsp_dec_rec_get` ([dsp_decrec_t](#) *dspdecrecInst*, double * *dpDestData*, long *lDestCount*)

6.2.1.349 [dsp_intfft_t](#) `dsp_int_fft_new` (void)

Creates new instance of FFT interpolator object.

Returns:

FFT interpolator instance

6.2.1.350 void `dsp_int_fft_delete` ([dsp_intfft_t](#))

Deletes FFT interpolator object instance.

Parameters:

dspintfftInst FFT interpolator instance

6.2.1.351 int `dsp_int_fft_initf` ([dsp_intfft_t](#) *dspintfftInst*, long *lIntFact*, long *lFiltSize*, int *iHighPass*)

[clFFTInterpolator::Initialize](#)

6.2.1.352 int `dsp_int_fft_init` ([dsp_intfft_t](#) *dspintfftInst*, long *lIntFact*, long *lFiltSize*, int *iHighPass*)

6.2.1.353 void `dsp_int_fft_uninit` ([dsp_intfft_t](#) *dspintfftInst*)

[clFFTInterpolator::Uninitialize](#)

6.2.1.354 void dsp_int_fft_putf ([dsp_intfft_t](#) *dspintfftInst*, const float * *fpSrcData*, long *lSrcCount*)

[clFFTInterpolator::Put](#)

6.2.1.355 void dsp_int_fft_put ([dsp_intfft_t](#) *dspintfftInst*, const double * *dpSrcData*, long *lSrcCount*)

6.2.1.356 int dsp_int_fft_getf ([dsp_intfft_t](#) *dspintfftInst*, float * *fpDestData*, long *lDestCount*)

[clFFTInterpolator::Get](#)

6.2.1.357 int dsp_int_fft_get ([dsp_intfft_t](#) *dspintfftInst*, double * *dpDestData*, long *lDestCount*)

6.2.1.358 [dsp_intfir_t](#) dsp_int_fir_new (void)

Creates new instance of FIR interpolator object.

Returns:

FIR interpolator instance

6.2.1.359 void dsp_int_fir_delete ([dsp_intfir_t](#))

Deletes FIR interpolator object instance.

Parameters:

dspintfirInst FIR interpolator instance

6.2.1.360 int dsp_int_fir_initf ([dsp_intfir_t](#) *dspintfirInst*, long *lIntFact*, int *iHighPass*)

[clFIRInterpolator::Initialize](#)

6.2.1.361 int dsp_int_fir_init ([dsp_intfir_t](#) *dspintfirInst*, long *lIntFact*, int *iHighPass*)

6.2.1.362 void dsp_int_fir_uninit ([dsp_intfir_t](#) *dspintfirInst*)

[clFIRInterpolator::Uninitialize](#)

6.2.1.363 void `dsp_int_fir_putf` ([dsp_intfir_t](#) *dspintfirInst*, const float * *fpSrcData*, long *lSrcCount*)

[cFIRInterpolator::Put](#)

6.2.1.364 void `dsp_int_fir_put` ([dsp_intfir_t](#) *dspintfirInst*, const double * *dpSrcData*, long *lSrcCount*)

6.2.1.365 int `dsp_int_fir_getf` ([dsp_intfir_t](#) *dspintfirInst*, float * *fpDestData*, long *lDestCount*)

[cFIRInterpolator::Get](#)

6.2.1.366 int `dsp_int_fir_get` ([dsp_intfir_t](#) *dspintfirInst*, double * *dpDestData*, long *lDestCount*)

6.2.1.367 [dsp_intiir_t](#) `dsp_int_iir_new` (void)

Creates new instance of IIR interpolator object.

Returns:

IIR interpolator instance

6.2.1.368 void `dsp_int_iir_delete` ([dsp_intiir_t](#))

Deletes IIR interpolator object instance.

Parameters:

dspintiirInst IIR interpolator instance

6.2.1.369 int `dsp_int_iir_initf` ([dsp_intiir_t](#) *dspintiirInst*, long *lIntFact*, int *iHighPass*)

[cIIRInterpolator::Initialize](#)

6.2.1.370 int `dsp_int_iir_init` ([dsp_intiir_t](#) *dspintiirInst*, long *lIntFact*, int *iHighPass*)

6.2.1.371 void `dsp_int_iir_uninit` ([dsp_intiir_t](#) *dspintiirInst*)

[cIIRInterpolator::Uninitialize](#)

6.2.1.372 void dsp_int_iir_putf ([dsp_intiir_t](#) *dspintiirInst*, const float * *fpSrcData*, long *lSrcCount*)

[clIIRInterpolator::Put](#)

6.2.1.373 void dsp_int_iir_put ([dsp_intiir_t](#) *dspintiirInst*, const double * *dpSrcData*, long *lSrcCount*)

6.2.1.374 int dsp_int_iir_getf ([dsp_intiir_t](#) *dspintiirInst*, float * *fpDestData*, long *lDestCount*)

[clIIRInterpolator::Get](#)

6.2.1.375 int dsp_int_iir_get ([dsp_intiir_t](#) *dspintiirInst*, double * *dpDestData*, long *lDestCount*)

6.2.1.376 [dsp_intrec_t](#) dsp_int_rec_new (void)

Creates new instance of recursive interpolator object.

Returns:

Recursive interpolator instance

6.2.1.377 void dsp_int_rec_delete ([dsp_intrec_t](#))

Deletes recursive interpolator object instance.

Parameters:

dspdecrecInst Recursive interpolator instance

6.2.1.378 int dsp_int_rec_initf ([dsp_intrec_t](#) *dspintrecInst*, long *lIntFact*, long *lFiltSize*, float *fBandCenter*, int *iFilterType*)

[clRecInterpolator::Initialize](#)

6.2.1.379 int dsp_int_rec_init ([dsp_intrec_t](#) *dspintrecInst*, long *lIntFact*, long *lFiltSize*, double *dBandCenter*, int *iFilterType*)

6.2.1.380 void dsp_int_rec_uninit ([dsp_intrec_t](#) *dspintrecInst*)

[clRecInterpolator::Uninitialize](#)

6.2.1.381 void `dsp_int_rec_putf` ([dsp_intrec_t](#) *dspintrecInst*, const float * *fpSrcData*, long *lSrcCount*)

[clRecInterpolator::Put](#)

6.2.1.382 void `dsp_int_rec_put` ([dsp_intrec_t](#) *dspintrecInst*, const double * *dpSrcData*, long *lSrcCount*)

6.2.1.383 int `dsp_int_rec_getf` ([dsp_intrec_t](#) *dspintrecInst*, float * *fpDestData*, long *lDestCount*)

[clRecInterpolator::Get](#)

6.2.1.384 int `dsp_int_rec_get` ([dsp_intrec_t](#) *dspintrecInst*, double * *dpDestData*, long *lDestCount*)

6.2.1.385 [dsp_filter_t](#) `dsp_filter_new` (void)

Creates new instance of FFT filter object.

Returns:

FFT filter instance

6.2.1.386 void `dsp_filter_delete` ([dsp_filter_t](#))

Deletes instance of FFT filter object.

Parameters:

dspfilterInst FFT filter instance

6.2.1.387 int `dsp_filter_initf` ([dsp_filter_t](#) *dspfilterInst*, long *lWindowSize*)

[clFilter::Initialize](#)

6.2.1.388 int `dsp_filter_init` ([dsp_filter_t](#) *dspfilterInst*, long *lWindowSize*)

6.2.1.389 int `dsp_filter_init2f` ([dsp_filter_t](#) *dspfilterInst*, long *lWindowSize*, const float * *fpFiltCoeffs*, float *fOverlap*, float *fBeta*)

6.2.1.390 int `dsp_filter_init2` ([dsp_filter_t](#) *dspfilterInst*, long *lWindowSize*, const double * *dpFiltCoeffs*, double *dOverlap*, double *dBeta*)

6.2.1.391 int `dsp_filter_init_lpf` ([dsp_filter_t](#) *dspfilterInst*, float *fPassBand*, float *fStopBand*, float *fRippleRatio*, float *fOverlap*)

[clFilter::InitializeLP](#)

6.2.1.392 `int dsp_filter_init_lp (dsp_filter_t dspfilterInst, double dPassBand, double dStopBand, double dRippleRatio, double dOverlap)`

6.2.1.393 `int dsp_filter_init_hpf (dsp_filter_t dspfilterInst, float fPassBand, float fStopBand, float fRippleRatio, float fOverlap)`

[clFilter::InitializeHP](#)

6.2.1.394 `int dsp_filter_init_hp (dsp_filter_t dspfilterInst, double dPassBand, double dStopBand, double dRippleRatio, double dOverlap)`

6.2.1.395 `void dsp_filter_set_coefssf (dsp_filter_t dspfilterInst, const float * fpFiltCoeffs)`

[clFilter::SetCoeffs](#)

6.2.1.396 `void dsp_filter_set_coefss (dsp_filter_t dspfilterInst, const double * dpFiltCoeffs)`

6.2.1.397 `void dsp_filter_set_ccoeffsf (dsp_filter_t dspfilterInst, const stpSCplx spFiltCoeffs, int iSmooth)`

6.2.1.398 `void dsp_filter_set_ccoeffs (dsp_filter_t dspfilterInst, const stpDCplx spFiltCoeffs, int iSmooth)`

6.2.1.399 `void dsp_filter_get_coefssf (dsp_filter_t dspfilterInst, float * fpFiltCoeffs)`

[clFilter::GetCoeffs](#)

6.2.1.400 `void dsp_filter_get_coefss (dsp_filter_t dspfilterInst, double * dpFiltCoeffs)`

6.2.1.401 `void dsp_filter_get_ccoeffsf (dsp_filter_t dspfilterInst, stpSCplx spFiltCoeffs)`

6.2.1.402 `void dsp_filter_get_ccoeffs (dsp_filter_t dspfilterInst, stpDCplx spFiltCoeffs)`

6.2.1.403 `void dsp_filter_putf (dsp_filter_t dspfilterInst, const float * fpSrcData, long iSrcCount)`

[clFilter::Put](#)

- 6.2.1.404 void `dsp_filter_put` ([dsp_filter_t](#) *dspfilterInst*, const double * *dpSrcData*, long *lSrcCount*)
- 6.2.1.405 void `dsp_filter_put2f` ([dsp_filter_t](#) *dspfilterInst*, const float * *fpSrcData*, long *lSrcCount*, const [stpSCplx](#) *spCoeffs*)
- 6.2.1.406 void `dsp_filter_put2` ([dsp_filter_t](#) *dspfilterInst*, const double * *dpSrcData*, long *lSrcCount*, const [stpDCplx](#) *spCoeffs*)
- 6.2.1.407 void `dsp_filter_getf` ([dsp_filter_t](#) *dspfilterInst*, float * *fpDestData*, long *lDestCount*)

[clFilter::Get](#)

- 6.2.1.408 void `dsp_filter_get` ([dsp_filter_t](#) *dspfilterInst*, double * *dpDestData*, long *lDestCount*)
- 6.2.1.409 void `dsp_filter_design_lpf` ([dsp_filter_t](#) *dspfilterInst*, float * *fpCorner*, int *iDCBlock*)

[clFilter::DesignLP](#)

- 6.2.1.410 void `dsp_filter_design_lp` ([dsp_filter_t](#) *dspfilterInst*, double * *dpCorner*, int *iDCBlock*)
- 6.2.1.411 void `dsp_filter_design_lp2f` ([dsp_filter_t](#) *dspfilterInst*, float * *fpCorner*, float *fSampleRate*, int *iDCBlock*)
- 6.2.1.412 void `dsp_filter_design_lp2` ([dsp_filter_t](#) *dspfilterInst*, double * *dpCorner*, double *dSampleRate*, int *iDCBlock*)
- 6.2.1.413 void `dsp_filter_design_hpf` ([dsp_filter_t](#) *dspfilterInst*, float * *fpCorner*)

[clFilter::DesignHP](#)

- 6.2.1.414 void `dsp_filter_design_hp` ([dsp_filter_t](#) *dspfilterInst*, double * *dpCorner*)
- 6.2.1.415 void `dsp_filter_design_hp2f` ([dsp_filter_t](#) *dspfilterInst*, float * *fpCorner*, float *fSampleRate*)
- 6.2.1.416 void `dsp_filter_design_hp2` ([dsp_filter_t](#) *dspfilterInst*, double * *dpCorner*, double *dSampleRate*)
- 6.2.1.417 void `dsp_filter_design_bpf` ([dsp_filter_t](#) *dspfilterInst*, float * *fpLowCorner*, float * *fpHighCorner*)

[clFilter::DesignBP](#)

- 6.2.1.418 void `dsp_filter_design_bp` ([dsp_filter_t](#) *dspfilterInst*, double * *dpLowCorner*, double * *dpHighCorner*)
- 6.2.1.419 void `dsp_filter_design_bp2f` ([dsp_filter_t](#) *dspfilterInst*, float * *fpLowCorner*, float * *fpHighCorner*, float *fSampleRate*)
- 6.2.1.420 void `dsp_filter_design_bp2` ([dsp_filter_t](#) *dspfilterInst*, double * *dpLowCorner*, double * *dpHighCorner*, double *dSampleRate*)
- 6.2.1.421 void `dsp_filter_design_brf` ([dsp_filter_t](#) *dspfilterInst*, float * *fpLowCorner*, float * *fpHighCorner*)

[clFilter::DesignBR](#)

- 6.2.1.422 void `dsp_filter_design_br` ([dsp_filter_t](#) *dspfilterInst*, double * *dpLowCorner*, double * *dpHighCorner*)
- 6.2.1.423 void `dsp_filter_design_br2f` ([dsp_filter_t](#) *dspfilterInst*, float * *fpLowCorner*, float * *fpHighCorner*, float *fSampleRate*)
- 6.2.1.424 void `dsp_filter_design_br2` ([dsp_filter_t](#) *dspfilterInst*, double * *dpLowCorner*, double * *dpHighCorner*, double *dSampleRate*)
- 6.2.1.425 [dsp_rebuf_t](#) `dsp_rebuf_new` (void)

Creates new instance of rebuffering object.

Returns:

Rebuffer instance

- 6.2.1.426 void `dsp_rebuf_delete` ([dsp_rebuf_t](#))

Deletes instance of rebuffering object.

Parameters:

dsprebufInst Rebuffer instance

6.2.1.427 void dsp_rebuf_putf ([dsp_rebuf_t](#) *dsprebufInst*, const float *
fpSrcData, long *lSrcCount*)

[clReBuffer::Put](#)

6.2.1.428 void dsp_rebuf_put ([dsp_rebuf_t](#) *dsprebufInst*, const double *
dpSrcData, long *lSrcCount*)

6.2.1.429 int dsp_rebuf_getf ([dsp_rebuf_t](#) *dsprebufInst*, float * *fpDestData*, long
lDestCount)

[clReBuffer::Get](#)

6.2.1.430 int dsp_rebuf_get ([dsp_rebuf_t](#) *dsprebufInst*, double * *dpDestData*,
long *lDestCount*)

6.2.1.431 long dsp_rebuf_size ([dsp_rebuf_t](#) *dsprebufInst*)

[clReBuffer::GetCount](#)

6.2.1.432 void dsp_rebuf_clear ([dsp_rebuf_t](#) *dsprebufInst*)

[clReBuffer::Clear](#)

6.2.1.433 void dsp_rebuf_copy ([dsp_rebuf_t](#) *dsprebufInst*, [dsp_rebuf_t](#)
dsprebufCopySrc)

[clReBuffer::operator=](#)

6.3 Compilers.hh File Reference

Defines

- `#define likely(x) (x)`
- `#define unlikely(x) (x)`
- `#define prefetch(x, w, l)`

6.3.1 Define Documentation

6.3.1.1 `#define likely(x) (x)`

6.3.1.2 `#define unlikely(x) (x)`

6.3.1.3 `#define prefetch(x, w, l)`

6.4 Condition.hh File Reference

```
#include <pthread.h>
#include <unistd.h>
#include <time.h>
#include <sys/time.h>
```

Classes

- class [clCondition](#)
Class implementation of POSIX condition variable.

6.5 DSPConfig.hh File Reference

Defines

- `#define DSP_FILT_DEF_OVERLAPF 0.75f`
- `#define DSP_FILT_DEF_OVERLAP 0.75`
- `#define DSP_FILT_DEF_BETAF 14.96454265f`
- `#define DSP_FILT_DEF_BETA 14.96454265`

6.5.1 Define Documentation

6.5.1.1 `#define DSP_FILT_DEF_OVERLAPF 0.75f`

6.5.1.2 `#define DSP_FILT_DEF_OVERLAP 0.75`

6.5.1.3 `#define DSP_FILT_DEF_BETAF 14.96454265f`

6.5.1.4 `#define DSP_FILT_DEF_BETA 14.96454265`

6.6 DSPOp.cc File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <limits.h>
#include <math.h>
#include <float.h>
#include "Compilers.hh"
#include "dsp/DSPOp.hh"
```

Functions

- `INLINE int FloatCompare (const void *vpValue1, const void *vpValue2)`
- `INLINE int DoubleCompare (const void *vpValue1, const void *vpValue2)`
- `INLINE int LongCompare (const void *vpValue1, const void *vpValue2)`

6.6.1 Function Documentation

6.6.1.1 `INLINE int FloatCompare (const void * vpValue1, const void * vpValue2)`

6.6.1.2 `INLINE int DoubleCompare (const void * vpValue1, const void * vpValue2)`

6.6.1.3 `INLINE int LongCompare (const void * vpValue1, const void * vpValue2)`

6.7 dspop.h File Reference

```
#include "dsp/dsptypes.h"
```

Typedefs

- typedef void * [dsp_t](#)
DSP object type.
- typedef void * [dsp_iircas_t](#)
Cascaded IIR filter object type.
- typedef void * [dsp_decfft_t](#)
FFT decimator object type.
- typedef void * [dsp_decfir_t](#)
FIR decimator object type.
- typedef void * [dsp_deciir_t](#)
IIR decimator object type.
- typedef void * [dsp_decrec_t](#)
Recursive decimator object type.
- typedef void * [dsp_intfft_t](#)
FFT interpolator object type.
- typedef void * [dsp_intfir_t](#)
FIR interpolator object type.
- typedef void * [dsp_intiir_t](#)
IIR interpolator object type.
- typedef void * [dsp_intrec_t](#)
Recursive interpolator object type.
- typedef void * [dsp_filter_t](#)
FFT filter object type.
- typedef void * [dsp_rebuf_t](#)
Rebuffering object type.

Functions

- void `dsp_init` (void)
Initialize DSP library.
- signed long `dsp_roundf` (float)
clDSPOp::Round
- signed long `dsp_round` (double)
- void `dsp_addf` (float *, float, long)
clDSPOp::Add
- void `dsp_add` (double *, double, long)
- void `dsp_caddf` (stpSCplx, stSCplx, long)
- void `dsp_cadd` (stpDCplx, stDCplx, long)
- void `dsp_add2f` (float *, const float *, long)
- void `dsp_add2` (double *, const double *, long)
- void `dsp_cadd2f` (stpSCplx, const stpSCplx, long)
- void `dsp_cadd2` (stpDCplx, const stpDCplx, long)
- void `dsp_add3f` (float *, const float *, const float *, long)
- void `dsp_add3` (double *, const double *, const double *, long)
- void `dsp_cadd3f` (stpSCplx, const stpSCplx, const stpSCplx, long)
- void `dsp_cadd3` (stpDCplx, const stpDCplx, const stpDCplx, long)
- void `dsp_subf` (float *, float, long)
clDSPOp::Sub
- void `dsp_sub` (double *, double, long)
- void `dsp_csubf` (stpSCplx, stSCplx, long)
- void `dsp_csub` (stpDCplx, stDCplx, long)
- void `dsp_sub2f` (float *, const float *, long)
- void `dsp_sub2` (double *, const double *, long)
- void `dsp_csub2f` (stpSCplx, const stpSCplx, long)
- void `dsp_csub2` (stpDCplx, const stpDCplx, long)
- void `dsp_sub3f` (float *, const float *, const float *, long)
- void `dsp_sub3` (double *, const double *, const double *, long)
- void `dsp_csub3f` (stpSCplx, const stpSCplx, const stpSCplx, long)
- void `dsp_csub3` (stpDCplx, const stpDCplx, const stpDCplx, long)
- void `dsp_mulf` (float *, float, long)
clDSPOp::Mul
- void `dsp_mul` (double *, double, long)
- void `dsp_chmulf` (stpSCplx, float, long)
- void `dsp_chmul` (stpDCplx, double, long)
- void `dsp_cmulf` (stpSCplx, stSCplx, long)
- void `dsp_cmul` (stpDCplx, stDCplx, long)
- void `dsp_mulf_nip` (float *, const float *, float, long)
- void `dsp_mul_nip` (double *, const double *, double, long)

- void `dsp_mul2f` (float *, const float *, long)
- void `dsp_mul2` (double *, const double *, long)
- void `dsp_chmul2f` (stpSCplx, const float *, long)
- void `dsp_chmul2` (stpDCplx, const double *, long)
- void `dsp_cmul2f` (stpSCplx, const stpSCplx, long)
- void `dsp_cmul2` (stpDCplx, const stpDCplx, long)
- void `dsp_mul3f` (float *, const float *, const float *, long)
- void `dsp_mul3` (double *, const double *, const double *, long)
- void `dsp_cmul3f` (stpSCplx, const stpSCplx, const stpSCplx, long)
- void `dsp_cmul3` (stpDCplx, const stpDCplx, const stpDCplx, long)
- void `dsp_ccmulf` (stpSCplx, const stpSCplx, long)
- void `dsp_ccmul` (stpDCplx, const stpDCplx, long)
- void `dsp_ccmulf_nip` (stpSCplx, const stpSCplx, const stpSCplx, long)
- void `dsp_ccmul_nip` (stpDCplx, const stpDCplx, const stpDCplx, long)
- void `dsp_divf` (float *, float, long)

clDSPOp::Div

- void `dsp_div` (double *, double, long)
- void `dsp_cdivf` (stpSCplx, stSCplx, long)
- void `dsp_cdiv` (stpDCplx, stDCplx, long)
- void `dsp_div2f` (float *, const float *, long)
- void `dsp_div2` (double *, const double *, long)
- void `dsp_cdiv2f` (stpSCplx, const stpSCplx, long)
- void `dsp_cdiv2` (stpDCplx, const stpDCplx, long)
- void `dsp_div3f` (float *, const float *, const float *, long)
- void `dsp_div3` (double *, const double *, const double *, long)
- void `dsp_cdiv3f` (stpSCplx, const stpSCplx, const stpSCplx, long)
- void `dsp_cdiv3` (stpDCplx, const stpDCplx, const stpDCplx, long)
- void `dsp_div1xf` (float *, long)

clDSPOp::Div1x

- void `dsp_div1x` (double *, long)
- void `dsp_div1xf_nip` (float *, const float *, long)
- void `dsp_div1x_nip` (double *, const double *, long)
- void `dsp_muladdf` (float *, float, float, long)

clDSPOp::MulAdd

- void `dsp_muladd` (double *, double, double, long)
- void `dsp_muladdf_nip` (float *, const float *, float, float, long)
- void `dsp_muladd_nip` (double *, const double *, double, double, long)
- void `dsp_absf` (float *, long)

clDSPOp::Abs

- void `dsp_abs` (double *, long)
- void `dsp_absf_nip` (float *, const float *, long)
- void `dsp_abs_nip` (double *, const double *, long)

- void [dsp_sqrtf](#) (float *, long)
clDSPOp::Sqrt
- void [dsp_sqrt](#) (double *, long)
- void [dsp_sqrtf_nip](#) (float *, const float *, long)
- void [dsp_sqrt_nip](#) (double *, const double *, long)
- void [dsp_zeroof](#) (float *, long)
clDSPOp::Zero
- void [dsp_zero](#) (double *, long)
- void [dsp_czeroof](#) (stpSCplx, long)
- void [dsp_czero](#) (stpDCplx, long)
- void [dsp_setf](#) (float *, float, long)
clDSPOp::Set
- void [dsp_set](#) (double *, double, long)
- void [dsp_csetf](#) (stpSCplx, stSCplx, long)
- void [dsp_cset](#) (stpDCplx, stDCplx, long)
- void [dsp_set2f](#) (float *, float, long, long, long)
- void [dsp_set2](#) (double *, double, long, long, long)
- void [dsp_cset2f](#) (stpSCplx, stSCplx, long, long, long)
- void [dsp_cset2](#) (stpDCplx, stDCplx, long, long, long)
- void [dsp_clipf](#) (float *, float, long)
clDSPOp::Clip
- void [dsp_clip](#) (double *, double, long)
- void [dsp_clipf_nip](#) (float *, const float *, float, long)
- void [dsp_clip_nip](#) (double *, const double *, double, long)
- void [dsp_clip2f](#) (float *, float, float, long)
- void [dsp_clip2](#) (double *, double, double, long)
- void [dsp_clip2f_nip](#) (float *, const float *, float, float, long)
- void [dsp_clip2_nip](#) (double *, const double *, double, double, long)
- void [dsp_clipzeroof](#) (float *, long)
clDSPOp::ClipZero
- void [dsp_clipzero](#) (double *, long)
- void [dsp_clipzeroof_nip](#) (float *, const float *, long)
- void [dsp_clipzero_nip](#) (double *, const double *, long)
- void [dsp_copyf](#) (float *, const float *, long)
clDSPOp::Copy
- void [dsp_copy](#) (double *, const double *, long)
- float [dsp_convolvef](#) (const float *, const float *, long)
clDSPOp::Convolve
- double [dsp_convolve](#) (const double *, const double *, long)

- void `dsp_convolve2f` (float *, const float *, const float *, long)
- void `dsp_convolve2` (double *, const double *, const double *, long)
- float `dsp_correlatef` (const float *, const float *, long)

clDSPOp::Correlate

- double `dsp_correlate` (const double *, const double *, long)
- void `dsp_correlate2f` (float *, const float *, const float *, long)
- void `dsp_correlate2` (double *, const double *, const double *, long)
- float `dsp_autocorr` (const float *, long)

clDSPOp::AutoCorrelate

- double `dsp_autocorr` (const double *, long)
- void `dsp_autocorr2f` (float *, const float *, long)
- void `dsp_autocorr2` (double *, const double *, long)
- float `dsp_dotproductf` (const float *, const float *, long)

clDSPOp::DotProduct

- double `dsp_dotproduct` (const double *, const double *, long)
- void `dsp_minmaxf` (float *, float *, const float *, long)

clDSPOp::MinMax

- void `dsp_minmax` (double *, double *, const double *, long)
- float `dsp_meanf` (const float *, long)

clDSPOp::Mean

- double `dsp_mean` (const double *, long)
- float `dsp_meadianf` (const float *, long)

clDSPOp::Median

- double `dsp_median` (const double *, long)
- void `dsp_negatef` (float *, long)

clDSPOp::Negate

- void `dsp_negate` (double *, long)
- void `dsp_negatef_nip` (float *, const float *, long)
- void `dsp_negate_nip` (double *, const double *, long)
- void `dsp_normalizef` (float *, long)

clDSPOp::Normalize

- void `dsp_normalize` (double *, long)
- void `dsp_normalizef_nip` (float *, const float *, long)
- void `dsp_normalize_nip` (double *, const double *, long)
- float `dsp_productf` (const float *, long)

clDSPOp::Product

- double `dsp_product` (const double *, long)

- void [dsp_reversef](#) (float *, long)
clDSPOp::Reverse
- void [dsp_reverse](#) (double *, long)
- void [dsp_reversef_nip](#) (float *, const float *, long)
- void [dsp_reverse_nip](#) (double *, const double *, long)
- void [dsp_scalef](#) (float *, long)
clDSPOp::Scale
- void [dsp_scale](#) (double *, long)
- void [dsp_scalef_nip](#) (float *, const float *, long)
- void [dsp_scale_nip](#) (double *, const double *, long)
- void [dsp_scale01f](#) (float *, long)
clDSPOp::Scale01
- void [dsp_scale01](#) (double *, long)
- void [dsp_scale01f_nip](#) (float *, const float *, long)
- void [dsp_Scale01_nip](#) (double *, const double *, long)
- void [dsp_sortf](#) (float *, long)
clDSPOp::Sort
- void [dsp_sort](#) (double *, long)
- void [dsp_sortl](#) (long *, long)
- void [dsp_stddevf](#) (float *, float *, const float *, long)
clDSPOp::StdDev
- void [dsp_stddev](#) (double *, double *, const double *, long)
- float [dsp_sumf](#) (const float *, long)
clDSPOp::Sum
- double [dsp_sum](#) (const double *, long)
- void [dsp_squaref](#) (float *, long)
clDSPOp::Square
- void [dsp_square](#) (double *, long)
- void [dsp_squaref_nip](#) (float *, const float *, long)
- void [dsp_square_nip](#) (double *, const double *, long)
- void [dsp_convertu8f](#) (float *, const unsigned char *, long)
clDSPOp::Convert
- void [dsp_convertu8](#) (double *, const unsigned char *, long)
- void [dsp_converts16f](#) (float *, const signed short *, long, int)
- void [dsp_converts16](#) (double *, const signed short *, long, int)
- void [dsp_converts32f](#) (float *, const signed int *, long, int)
- void [dsp_converts32](#) (double *, const signed int *, long, int)
- void [dsp_converttd64f](#) (float *, const double *, long)

$$clDSPOp::CartToPolar$$
$$clDSPOp::PolarToCart$$
 $clDSPOp::CrossCorr$

clDSPOp::DelCrossCorr

$$clDSPOp::Energy$$

clDSPOp::Magnitude

- void `dsp_powerf` (float *, const `stpSCplx`, long)
clDSPOp::Power
- void `dsp_power` (double *, const `stpDCplx`, long)
- void `dsp_phasef` (float *, const `stpSCplx`, long)
clDSPOp::Phase
- void `dsp_phase` (double *, const `stpDCplx`, long)
- void `dsp_powerphasef` (float *, float *, const `stpSCplx`, long)
clDSPOp::PowerPhase
- void `dsp_powerphase` (double *, double *, const `stpDCplx`, long)
- void `dsp_decimatef` (float *, const float *, long, long)
clDSPOp::Decimate
- void `dsp_decimate` (double *, const double *, long, long)
- void `dsp_decimateavgf` (float *, const float *, long, long)
clDSPOp::DecimateAvg
- void `dsp_decimateavg` (double *, const double *, long, long)
- void `dsp_interpolatef` (float *, const float *, long, long)
clDSPOp::Interpolate
- void `dsp_interpolate` (double *, const double *, long, long)
- void `dsp_interpolateavgf` (float *, const float *, long, long)
clDSPOp::InterpolateAvg
- void `dsp_interpolateavg` (double *, const double *, long, long)
- void `dsp_resamplef` (float *, long, const float *, long)
clDSPOp::Resample
- void `dsp_resample` (double *, long, const double *, long)
- void `dsp_resampleavgf` (float *, long, const float *, long)
clDSPOp::ResampleAvg
- void `dsp_resampleavg` (double *, long, const double *, long)
- float `dsp_rmsf` (const float *, long)
clDSPOp::RMS
- double `dsp_rms` (const double *, long)
- float `dsp_variancef` (float *, float *, const float *, long)
clDSPOp::Variance
- double `dsp_variance` (double *, double *, const double *, long)
- float `dsp_peaklevelf` (const float *fpSrc, long)
clDSPOp::PeakLevel

- double [dsp_peaklevel](#) (const double *dpSrc, long)
- void [dsp_mixf](#) (float *, const float *, long)
clDSPOp::Mix
- void [dsp_mix](#) (double *, const double *, long)
- void [dsp_mix2f](#) (float *, const float *, const float *, long)
- void [dsp_mix2](#) (double *, const double *, const double *, long)
- void [dsp_mixnf](#) (float *, const float *, long, long)
- void [dsp_mixn](#) (double *, const double *, long, long)
- void [dsp_extractf](#) (float *, const float *, long, long, long)
clDSPOp::Extract
- void [dsp_extract](#) (double *, const double *, long, long, long)
- void [dsp_packf](#) (float *, const float *, long, long, long)
clDSPOp::Pack
- void [dsp_pack](#) (double *, const double *, long, long, long)
- void [dsp_fftw_convertf2cf](#) (stpSCplx, const float *, long)
clDSPOp::FFTWConvert
- void [dsp_fftw_convertf2cd](#) (stpDCplx, const float *, long)
- void [dsp_fftw_convertf2cf](#) (stpSCplx, const double *, long)
- void [dsp_fftw_convertf2cd](#) (stpDCplx, const double *, long)
- void [dsp_fftw_convertcf2f](#) (float *, const stpSCplx, long)
- void [dsp_fftw_convertcd2f](#) (float *, const stpDCplx, long)
- void [dsp_fftw_convertcf2d](#) (double *, const stpSCplx, long)
- void [dsp_fftw_convertcd2d](#) (double *, const stpDCplx, long)
- [dsp_t dsp_new](#) (void)
Creates new instance of DSP object.
- void [dsp_delete](#) (dsp_t)
Deletes DSP object instance.
- void [dsp_win_bartlett](#) (dsp_t, float *, long)
clDSPOp::WinBartlett
- void [dsp_win_bartlett](#) (dsp_t, double *, long)
- void [dsp_win_blackmanf](#) (dsp_t, float *, long, float)
clDSPOp::WinBlackman
- void [dsp_win_blackman](#) (dsp_t, double *, long, double)
- void [dsp_win_blackman_harrisf](#) (dsp_t, float *, long)
clDSPOp::WinBlackmanHarris
- void [dsp_win_blackman_harris](#) (dsp_t, double *, long)

- void [dsp_win_cos_taperedf](#) (dsp_t, float *, long)
clDSPOp::WinCosTapered
- void [dsp_win_cos_tapered](#) (dsp_t, double *, long)
- void [dsp_win_exact_blackmanf](#) (dsp_t, float *, long)
clDSPOp::WinExactBlackman
- void [dsp_win_exact_blackman](#) (dsp_t, double *, long)
- void [dsp_win_expf](#) (dsp_t, float *, float, long)
clDSPOp::WinExp
- void [dsp_win_exp](#) (dsp_t, double *, double, long)
- void [dsp_win_flat_topf](#) (dsp_t, float *, long)
clDSPOp::WinFlatTop
- void [dsp_win_flat_top](#) (dsp_t, double *, long)
- void [dsp_win_generic_cosf](#) (dsp_t, float *, long, const float *, long)
clDSPOp::WinGenericCos
- void [dsp_win_generic_cos](#) (dsp_t, double *, long, const double *, long)
- void [dsp_win_hammingf](#) (dsp_t, float *, long)
clDSPOp::WinHamming
- void [dsp_win_hamming](#) (dsp_t, double *, long)
- void [dsp_win_hanningf](#) (dsp_t, float *, long)
clDSPOp::WinHanning
- void [dsp_win_hanning](#) (dsp_t, double *, long)
- void [dsp_win_kaiserf](#) (dsp_t, float *, float, long)
clDSPOp::WinKaiser
- void [dsp_win_kaiser](#) (dsp_t, double *, double, long)
- void [dsp_win_kaiser_besself](#) (dsp_t, float *, float, long)
clDSPOp::WinKaiserBessel
- void [dsp_win_kaiser_bessel](#) (dsp_t, double *, double, long)
- void [dsp_win_tukeyf](#) (dsp_t, float *, long)
clDSPOp::WinTukey
- void [dsp_win_tukey](#) (dsp_t, double *, long)
- void [dsp_win_dolph_chebyshevf](#) (dsp_t, float *, float, long)
clDSPOp::WinDolphChebyshev
- void [dsp_win_dolph_chebyshev](#) (dsp_t, double *, double, long)
- long [dsp_rebufferf](#) (dsp_t, float *, const float *, long, long)
clDSPOp::ReBuffer

- long `dsp_rebuffer` (`dsp_t`, double *, const double *, long, long)
- float `dsp_deg2radf` (`dsp_t`, float)
`clDSPOp::DegToRad`
- double `dsp_deg2rad` (`dsp_t`, double)
- float `dsp_rad2degf` (`dsp_t`, float)
`clDSPOp::RadToDeg`
- double `dsp_rad2deg` (`dsp_t`, double)
- void `dsp_fir_allocatef` (`dsp_t`, const float *, long)
`clDSPOp::FIRAllocate`
- void `dsp_fir_allocate` (`dsp_t`, const double *, long)
- void `dsp_fir_free` (`dsp_t`)
`clDSPOp::FIRFree`
- void `dsp_fir_filterf` (`dsp_t`, float *, long)
`clDSPOp::FIRFilter`
- void `dsp_fir_filter` (`dsp_t`, double *, long)
- void `dsp_fir_filterf_nip` (`dsp_t`, float *, const float *, long)
- void `dsp_fir_filter_nip` (`dsp_t`, double *, const double *, long)
- void `dsp_fir_filterf_fst` (`dsp_t`, float *, float *, long)
`clDSPOp::FIRFilterF`
- void `dsp_fir_filter_fst` (`dsp_t`, double *, double *, long)
- void `dsp_iir_initf` (`dsp_t`, const float *)
`clDSPOp::IIRInitialize`
- void `dsp_iir_init` (`dsp_t`, const double *)
- void `dsp_iir_filterf` (`dsp_t`, float *, long)
`clDSPOp::IIRFilter`
- void `dsp_iir_filter` (`dsp_t`, double *, long)
- void `dsp_iir_filterf_nip` (`dsp_t`, float *, const float *, long)
- void `dsp_iir_filter_nip` (`dsp_t`, double *, const double *, long)
- void `dsp_iir_clear` (`dsp_t`)
`clDSPOp::IIRClear`
- void `dsp_fft_init` (`dsp_t`, long, int)
`clDSPOp::FFTInitialize`
- void `dsp_fft_uninit` (`dsp_t`)
`clDSPOp::FFTUninitialize`

- void `dsp_fftf` (`dsp_t`, `stpSCplx`, `float *`)
clDSPOp::FFTi
- void `dsp_fft` (`dsp_t`, `stpDCplx`, `double *`)
- void `dsp_fftf_nip` (`dsp_t`, `stpSCplx`, `const float *`)
clDSPOp::FFTo
- void `dsp_fft_nip` (`dsp_t`, `stpDCplx`, `const double *`)
- void `dsp_cfftf_nip` (`dsp_t`, `stpSCplx`, `const stpSCplx`)
- void `dsp_cfft_nip` (`dsp_t`, `stpDCplx`, `const stpDCplx`)
- void `dsp_ifftf_nip` (`dsp_t`, `float *`, `const stpSCplx`)
clDSPOp::IFFTo
- void `dsp_ifft_nip` (`dsp_t`, `double *`, `const stpDCplx`)
- void `dsp_cifftf_nip` (`dsp_t`, `stpSCplx`, `const stpSCplx`)
- void `dsp_cifft_nip` (`dsp_t`, `stpDCplx`, `const stpDCplx`)
- `dsp_iircas_t dsp_iir_cas_new` (`void`)
Creates new instance of cascaded IIR object.
- void `dsp_iir_cas_delete` (`dsp_iircas_t`)
Deletes cascaded IIR object instance.
- int `dsp_iir_cas_initf` (`dsp_iircas_t`, `const float[][5]`, `long`)
clIIRCascade::Initialize
- int `dsp_iir_cas_init` (`dsp_iircas_t`, `const double[][5]`, `long`)
- void `dsp_iir_cas_uninit` (`dsp_iircas_t`)
- void `dsp_iir_cas_processf` (`dsp_iircas_t`, `float *`, `long`)
- void `dsp_iir_cas_process` (`dsp_iircas_t`, `double *`, `long`)
- void `dsp_iir_cas_processf_nip` (`dsp_iircas_t`, `float *`, `const float *`, `long`)
- void `dsp_iir_cas_process_nip` (`dsp_iircas_t`, `double *`, `const double *`, `long`)
- void `dsp_iir_cas_clear` (`dsp_iircas_t`)
- `dsp_decfft_t dsp_dec_fft_new` (`void`)
Creates new instance of FFT decimator object.
- void `dsp_dec_fft_delete` (`dsp_decfft_t`)
Deletes FFT decimator object instance.
- int `dsp_dec_fft_initf` (`dsp_decfft_t`, `long`, `long`, `int`)
clFFTDecimator::Initialize
- int `dsp_dec_fft_init` (`dsp_decfft_t`, `long`, `long`, `int`)
- void `dsp_dec_fft_uninit` (`dsp_decfft_t`)
clFFTDecimator::Uninitialize
- void `dsp_dec_fft_putf` (`dsp_decfft_t`, `const float *`, `long`)

clFFTDecimator::Put

- void `dsp_dec_fft_put` (`dsp_decffft_t`, const double *, long)
- int `dsp_dec_fft_getf` (`dsp_decffft_t`, float *, long)

clFFTDecimator::Get

- int `dsp_dec_fft_get` (`dsp_decffft_t`, double *, long)
- `dsp_decfir_t dsp_dec_fir_new` (void)

Creates new instance of FIR decimator object.

- void `dsp_dec_fir_delete` (`dsp_decfir_t`)

Deletes FIR decimator object instance.

- int `dsp_dec_fir_initf` (`dsp_decfir_t`, long, int)

clFIRDecimator::Initialize

- int `dsp_dec_fir_init` (`dsp_decfir_t`, long, int)
- void `dsp_dec_fir_uninit` (`dsp_decfir_t`)

clFIRDecimator::Uninitialize

- void `dsp_dec_fir_putf` (`dsp_decfir_t`, const float *, long)

clFIRDecimator::Put

- void `dsp_dec_fir_put` (`dsp_decfir_t`, const double *, long)
- int `dsp_dec_fir_getf` (`dsp_decfir_t`, float *, long)

clFIRDecimator::Get

- int `dsp_dec_fir_get` (`dsp_decfir_t`, double *, long)
- `dsp_deciir_t dsp_dec_iir_new` (void)

Creates new instance of IIR decimator object.

- void `dsp_dec_iir_delete` (`dsp_deciir_t`)

Deletes IIR decimator object instance.

- int `dsp_dec_iir_initf` (`dsp_deciir_t`, long, int)

clIIRDecimator::Initialize

- int `dsp_dec_iir_init` (`dsp_deciir_t`, long, int)
- void `dsp_dec_iir_uninit` (`dsp_deciir_t`)

clIIRDecimator::Uninitialize

- void `dsp_dec_iir_putf` (`dsp_deciir_t`, const float *, long)

clIIRDecimator::Put

- void `dsp_dec_iir_put` (`dsp_deciir_t`, const double *, long)
- int `dsp_dec_iir_getf` (`dsp_deciir_t`, float *, long)

clIIRDecimator::Get

- int [dsp_dec_iir_get](#) (dsp_deciir_t, double *, long)
- [dsp_decrec_t dsp_dec_rec_new](#) (void)
Creates new instance of recursive decimator object.
- void [dsp_dec_rec_delete](#) (dsp_decrec_t)
Deletes recursive decimator object instance.
- int [dsp_dec_rec_initf](#) (dsp_decrec_t, long, long, float, int)
clRecDecimator::Initialize
- int [dsp_dec_rec_init](#) (dsp_decrec_t, long, long, double, int)
- void [dsp_dec_rec_uninit](#) (dsp_decrec_t)
clRecDecimator::Uninitialize
- void [dsp_dec_rec_putf](#) (dsp_decrec_t, const float *, long)
clRecDecimator::Put
- void [dsp_dec_rec_put](#) (dsp_decrec_t, const double *, long)
- int [dsp_dec_rec_getf](#) (dsp_decrec_t, float *, long)
clRecDecimator::Get
- int [dsp_dec_rec_get](#) (dsp_decrec_t, double *, long)
- [dsp_intfft_t dsp_int_fft_new](#) (void)
Creates new instance of FFT interpolator object.
- void [dsp_int_fft_delete](#) (dsp_intfft_t)
Deletes FFT interpolator object instance.
- int [dsp_int_fft_initf](#) (dsp_intfft_t, long, long, int)
clFFTInterpolator::Initialize
- int [dsp_int_fft_init](#) (dsp_intfft_t, long, long, int)
- void [dsp_int_fft_uninit](#) (dsp_intfft_t)
clFFTInterpolator::Uninitialize
- void [dsp_int_fft_putf](#) (dsp_intfft_t, const float *, long)
clFFTInterpolator::Put
- void [dsp_int_fft_put](#) (dsp_intfft_t, const double *, long)
- int [dsp_int_fft_getf](#) (dsp_intfft_t, float *, long)
clFFTInterpolator::Get
- int [dsp_int_fft_get](#) (dsp_intfft_t, double *, long)
- [dsp_intfir_t dsp_int_fir_new](#) (void)

Creates new instance of FIR interpolator object.

- void `dsp_int_fir_delete` (`dsp_intfir_t`)
Deletes FIR interpolator object instance.
- int `dsp_int_fir_initf` (`dsp_intfir_t`, long, int)
clFIRInterpolator::Initialize
- int `dsp_int_fir_init` (`dsp_intfir_t`, long, int)
- void `dsp_int_fir_uninit` (`dsp_intfir_t`)
clFIRInterpolator::Uninitialize
- void `dsp_int_fir_putf` (`dsp_intfir_t`, const float *, long)
clFIRInterpolator::Put
- void `dsp_int_fir_put` (`dsp_intfir_t`, const double *, long)
- int `dsp_int_fir_getf` (`dsp_intfir_t`, float *, long)
clFIRInterpolator::Get
- int `dsp_int_fir_get` (`dsp_intfir_t`, double *, long)
- `dsp_intiir_t` `dsp_int_iir_new` (void)
Creates new instance of IIR interpolator object.
- void `dsp_int_iir_delete` (`dsp_intiir_t`)
Deletes IIR interpolator object instance.
- int `dsp_int_iir_initf` (`dsp_intiir_t`, long, int)
clIIRInterpolator::Initialize
- int `dsp_int_iir_init` (`dsp_intiir_t`, long, int)
- void `dsp_int_iir_uninit` (`dsp_intiir_t`)
clIIRInterpolator::Uninitialize
- void `dsp_int_iir_putf` (`dsp_intiir_t`, const float *, long)
clIIRInterpolator::Put
- void `dsp_int_iir_put` (`dsp_intiir_t`, const double *, long)
- int `dsp_int_iir_getf` (`dsp_intiir_t`, float *, long)
clIIRInterpolator::Get
- int `dsp_int_iir_get` (`dsp_intiir_t`, double *, long)
- `dsp_intrec_t` `dsp_int_rec_new` (void)
Creates new instance of recursive interpolator object.
- void `dsp_int_rec_delete` (`dsp_intrec_t`)
Deletes recursive interpolator object instance.

- int `dsp_int_rec_initf` (`dsp_intrec_t`, long, long, float, int)
clRecInterpolator::Initialize
- int `dsp_int_rec_init` (`dsp_intrec_t`, long, long, double, int)
- void `dsp_int_rec_uninit` (`dsp_intrec_t`)
clRecInterpolator::Uninitialize
- void `dsp_int_rec_putf` (`dsp_intrec_t`, const float *, long)
clRecInterpolator::Put
- void `dsp_int_rec_put` (`dsp_intrec_t`, const double *, long)
- int `dsp_int_rec_getf` (`dsp_intrec_t`, float *, long)
clRecInterpolator::Get
- int `dsp_int_rec_get` (`dsp_intrec_t`, double *, long)
- `dsp_filter_t` `dsp_filter_new` (void)
Creates new instance of FFT filter object.
- void `dsp_filter_delete` (`dsp_filter_t`)
Deletes instance of FFT filter object.
- int `dsp_filter_initf` (`dsp_filter_t`, long)
clFilter::Initialize
- int `dsp_filter_init` (`dsp_filter_t`, long)
- int `dsp_filter_init2f` (`dsp_filter_t`, long, const float *, float, float)
- int `dsp_filter_init2` (`dsp_filter_t`, long, const double *, double, double)
- int `dsp_filter_init_lpf` (`dsp_filter_t`, float, float, float, float)
clFilter::InitializeLP
- int `dsp_filter_init_lp` (`dsp_filter_t`, double, double, double, double)
- int `dsp_filter_init_hpf` (`dsp_filter_t`, float, float, float, float)
clFilter::InitializeHP
- int `dsp_filter_init_hp` (`dsp_filter_t`, double, double, double, double)
- void `dsp_filter_set_coeffs` (`dsp_filter_t`, const float *)
clFilter::SetCoeffs
- void `dsp_filter_set_ceoffs` (`dsp_filter_t`, const double *)
- void `dsp_filter_set_ccoeffsf` (`dsp_filter_t`, const `stpSCplx`, int)
- void `dsp_filter_set_ccoeffs` (`dsp_filter_t`, const `stpDCplx`, int)
- void `dsp_filter_get_ccoeffsf` (`dsp_filter_t`, float *)
clFilter::GetCoeffs
- void `dsp_filter_get_coeffs` (`dsp_filter_t`, double *)

- void `dsp_filter_get_ccoeffsf` (`dsp_filter_t`, `stpSCplx`)
- void `dsp_filter_get_ccoeffs` (`dsp_filter_t`, `stpDCplx`)
- void `dsp_filter_putf` (`dsp_filter_t`, const float *, long)
clFilter::Put
- void `dsp_filter_put` (`dsp_filter_t`, const double *, long)
- void `dsp_filter_put2f` (`dsp_filter_t`, const float *, long, const `stpSCplx`)
- void `dsp_filter_put2` (`dsp_filter_t`, const double *, long, const `stpDCplx`)
- void `dsp_filter_getf` (`dsp_filter_t`, float *, long)
clFilter::Get
- void `dsp_filter_get` (`dsp_filter_t`, double *, long)
- void `dsp_filter_design_lpf` (`dsp_filter_t`, float *, int)
clFilter::DesignLP
- void `dsp_filter_design_lp` (`dsp_filter_t`, double *, int)
- void `dsp_filter_design_lp2f` (`dsp_filter_t`, float *, float, int)
- void `dsp_filter_design_lp2` (`dsp_filter_t`, double *, double, int)
- void `dsp_filter_design_hpf` (`dsp_filter_t`, float *)
clFilter::DesignHP
- void `dsp_filter_design_hp` (`dsp_filter_t`, double *)
- void `dsp_filter_design_hp2f` (`dsp_filter_t`, float *, float)
- void `dsp_filter_design_hp2` (`dsp_filter_t`, double *, double)
- void `dsp_filter_design_bpf` (`dsp_filter_t`, float *, float *)
clFilter::DesignBP
- void `dsp_filter_design_bp` (`dsp_filter_t`, double *, double *)
- void `dsp_filter_design_bp2f` (`dsp_filter_t`, float *, float *, float)
- void `dsp_filter_design_bp2` (`dsp_filter_t`, double *, double *, double)
- void `dsp_filter_design_br` (`dsp_filter_t`, float *, float *)
clFilter::DesignBR
- void `dsp_filter_design_br` (`dsp_filter_t`, double *, double *)
- void `dsp_filter_design_br2f` (`dsp_filter_t`, float *, float *, float)
- void `dsp_filter_design_br2` (`dsp_filter_t`, double *, double *, double)
- `dsp_rebuf_t dsp_rebuf_new` (void)
Creates new instance of rebuffering object.
- void `dsp_rebuf_delete` (`dsp_rebuf_t`)
Deletes instance of rebuffering object.
- void `dsp_rebuf_putf` (`dsp_rebuf_t`, const float *, long)
clReBuffer::Put
- void `dsp_rebuf_put` (`dsp_rebuf_t`, const double *, long)

- int `dsp_rebuf_getf` (`dsp_rebuf_t`, float *, long)
clReBuffer::Get
- int `dsp_rebuf_get` (`dsp_rebuf_t`, double *, long)
- long `dsp_rebuf_size` (`dsp_rebuf_t`)
clReBuffer::GetCount
- void `dsp_rebuf_clear` (`dsp_rebuf_t`)
clReBuffer::Clear
- void `dsp_rebuf_copy` (`dsp_rebuf_t`, `dsp_rebuf_t`)
clReBuffer::operator=

6.7.1 Typedef Documentation

6.7.1.1 typedef void* `dsp_t`

DSP object type.

6.7.1.2 typedef void* `dsp_iircas_t`

Cascaded IIR filter object type.

6.7.1.3 typedef void* `dsp_decfft_t`

FFT decimator object type.

6.7.1.4 typedef void* `dsp_decfir_t`

FIR decimator object type.

6.7.1.5 typedef void* `dsp_deciir_t`

IIR decimator object type.

6.7.1.6 typedef void* `dsp_decrec_t`

Recursive decimator object type.

6.7.1.7 typedef void* `dsp_intfft_t`

FFT interpolator object type.

6.7.1.8 typedef void* [dsp_intfir_t](#)

FIR interpolator object type.

6.7.1.9 typedef void* [dsp_intiir_t](#)

IIR interpolator object type.

6.7.1.10 typedef void* [dsp_intrec_t](#)

Recursive interpolator object type.

6.7.1.11 typedef void* [dsp_filter_t](#)

FFT filter object type.

6.7.1.12 typedef void* [dsp_rebuf_t](#)

Rebuffering object type.

6.7.2 Function Documentation**6.7.2.1** void [dsp_init](#) (void)

Initialize DSP library.

This should be called first to enable possible hardware dependent optimizations.

6.7.2.2 signed long [dsp_roundf](#) (float)

[clDSPOp::Round](#)

6.7.2.3 signed long [dsp_round](#) (double)**6.7.2.4** void [dsp_addf](#) (float *, float, long)

[clDSPOp::Add](#)

6.7.2.5 void dsp_add (double *, double, long)

6.7.2.6 void dsp_caddf ([stpSCplx](#), [stSCplx](#), long)

6.7.2.7 void dsp_cadd ([stpDCplx](#), [stDCplx](#), long)

6.7.2.8 void dsp_add2f (float *, const float *, long)

6.7.2.9 void dsp_add2 (double *, const double *, long)

6.7.2.10 void dsp_cadd2f ([stpSCplx](#), const *stpSCplx*, long)

6.7.2.11 void dsp_cadd2 ([stpDCplx](#), const *stpDCplx*, long)

6.7.2.12 void dsp_add3f (float *, const float *, const float *, long)

6.7.2.13 void dsp_add3 (double *, const double *, const double *, long)

6.7.2.14 void dsp_cadd3f ([stpSCplx](#), const *stpSCplx*, const *stpSCplx*, long)

6.7.2.15 void dsp_cadd3 ([stpDCplx](#), const *stpDCplx*, const *stpDCplx*, long)

6.7.2.16 void dsp_subf (float *, float, long)

[clDSPOp::Sub](#)

6.7.2.17 `void dsp_sub (double *, double, long)`

6.7.2.18 `void dsp_csubf (stpSCplx, stSCplx, long)`

6.7.2.19 `void dsp_csub (stpDCplx, stDCplx, long)`

6.7.2.20 `void dsp_sub2f (float *, const float *, long)`

6.7.2.21 `void dsp_sub2 (double *, const double *, long)`

6.7.2.22 `void dsp_csub2f (stpSCplx, const stpSCplx, long)`

6.7.2.23 `void dsp_csub2 (stpDCplx, const stpDCplx, long)`

6.7.2.24 `void dsp_sub3f (float *, const float *, const float *, long)`

6.7.2.25 `void dsp_sub3 (double *, const double *, const double *, long)`

6.7.2.26 `void dsp_csub3f (stpSCplx, const stpSCplx, const stpSCplx, long)`

6.7.2.27 `void dsp_csub3 (stpDCplx, const stpDCplx, const stpDCplx, long)`

6.7.2.28 `void dsp_mulf (float *, float, long)`

[clDSPOp::Mul](#)

- 6.7.2.29 void dsp_mul (double *, double, long)
- 6.7.2.30 void dsp_chmulf ([stpSCplx](#), float, long)
- 6.7.2.31 void dsp_chmul ([stpDCplx](#), double, long)
- 6.7.2.32 void dsp_cmulf ([stpSCplx](#), [stSCplx](#), long)
- 6.7.2.33 void dsp_cmul ([stpDCplx](#), [stDCplx](#), long)
- 6.7.2.34 void dsp_mulf_nip (float *, const float *, float, long)
- 6.7.2.35 void dsp_mul_nip (double *, const double *, double, long)
- 6.7.2.36 void dsp_mul2f (float *, const float *, long)
- 6.7.2.37 void dsp_mul2 (double *, const double *, long)
- 6.7.2.38 void dsp_chmul2f ([stpSCplx](#), const float *, long)
- 6.7.2.39 void dsp_chmul2 ([stpDCplx](#), const double *, long)
- 6.7.2.40 void dsp_cmulf2 ([stpSCplx](#), const *stpSCplx*, long)
- 6.7.2.41 void dsp_cmul2 ([stpDCplx](#), const *stpDCplx*, long)
- 6.7.2.42 void dsp_mul3f (float *, const float *, const float *, long)
- 6.7.2.43 void dsp_mul3 (double *, const double *, const double *, long)
- 6.7.2.44 void dsp_cmulf3 ([stpSCplx](#), const *stpSCplx*, const *stpSCplx*, long)
- 6.7.2.45 void dsp_cmul3 ([stpDCplx](#), const *stpDCplx*, const *stpDCplx*, long)
- 6.7.2.46 void dsp_ccmulf ([stpSCplx](#), const *stpSCplx*, long)
- 6.7.2.47 void dsp_ccmul ([stpDCplx](#), const *stpDCplx*, long)
- 6.7.2.48 void dsp_ccmulf_nip ([stpSCplx](#), const *stpSCplx*, const *stpSCplx*, long)
- 6.7.2.49 void dsp_ccmul_nip ([stpDCplx](#), const *stpDCplx*, const *stpDCplx*, long)
- 6.7.2.50 void dsp_divf (float *, float, long)

[clDSPOp::Div](#)

- 6.7.2.51 void dsp_div (double *, double, long)
- 6.7.2.52 void dsp_cdivf ([stpSCplx](#), [stSCplx](#), long)
- 6.7.2.53 void dsp_cdiv ([stpDCplx](#), [stDCplx](#), long)
- 6.7.2.54 void dsp_div2f (float *, const float *, long)
- 6.7.2.55 void dsp_div2 (double *, const double *, long)
- 6.7.2.56 void dsp_cdiv2f ([stpSCplx](#), const *stpSCplx*, long)
- 6.7.2.57 void dsp_cdiv2 ([stpDCplx](#), const *stpDCplx*, long)
- 6.7.2.58 void dsp_div3f (float *, const float *, const float *, long)
- 6.7.2.59 void dsp_div3 (double *, const double *, const double *, long)
- 6.7.2.60 void dsp_cdiv3f ([stpSCplx](#), const *stpSCplx*, const *stpSCplx*, long)
- 6.7.2.61 void dsp_cdiv3 ([stpDCplx](#), const *stpDCplx*, const *stpDCplx*, long)
- 6.7.2.62 void dsp_div1xf (float *, long)

[cIDSPOp::Div1x](#)

- 6.7.2.63 void dsp_div1x (double *, long)
- 6.7.2.64 void dsp_div1xf_nip (float *, const float *, long)
- 6.7.2.65 void dsp_div1x_nip (double *, const double *, long)
- 6.7.2.66 void dsp_muladdf (float *, float, float, long)

[cIDSPOp::MulAdd](#)

- 6.7.2.67 void dsp_muladd (double *, double, double, long)
- 6.7.2.68 void dsp_muladdf_nip (float *, const float *, float, float, long)
- 6.7.2.69 void dsp_muladd_nip (double *, const double *, double, double, long)
- 6.7.2.70 void dsp_absf (float *, long)

[cIDSPOp::Abs](#)

6.7.2.71 void dsp_abs (double *, long)

6.7.2.72 void dsp_absf_nip (float *, const float *, long)

6.7.2.73 void dsp_abs_nip (double *, const double *, long)

6.7.2.74 void dsp_sqrtf (float *, long)

[clDSPOp::Sqrt](#)

6.7.2.75 void dsp_sqrt (double *, long)

6.7.2.76 void dsp_sqrtf_nip (float *, const float *, long)

6.7.2.77 void dsp_sqrt_nip (double *, const double *, long)

6.7.2.78 void dsp_zerof (float *, long)

[clDSPOp::Zero](#)

6.7.2.79 void dsp_zero (double *, long)

6.7.2.80 void dsp_czerof ([stpSCplx](#), long)

6.7.2.81 void dsp_czero ([stpDCplx](#), long)

6.7.2.82 void dsp_setf (float *, float, long)

[clDSPOp::Set](#)

6.7.2.83 void dsp_set (double *, double, long)

6.7.2.84 void dsp_csetf ([stpSCplx](#), [stSCplx](#), long)

6.7.2.85 void dsp_cset ([stpDCplx](#), [stDCplx](#), long)

6.7.2.86 void dsp_set2f (float *, float, long, long, long)

6.7.2.87 void dsp_set2 (double *, double, long, long, long)

6.7.2.88 void dsp_cset2f ([stpSCplx](#), [stSCplx](#), long, long, long)

6.7.2.89 void dsp_cset2 ([stpDCplx](#), [stDCplx](#), long, long, long)

6.7.2.90 void dsp_clipf (float *, float, long)

[clDSPOp::Clip](#)

6.7.2.91 void dsp_clip (double *, double, long)

6.7.2.92 void dsp_clipf_nip (float *, const float *, float, long)

6.7.2.93 void dsp_clip_nip (double *, const double *, double, long)

6.7.2.94 void dsp_clip2f (float *, float, float, long)

6.7.2.95 void dsp_clip2 (double *, double, double, long)

6.7.2.96 void dsp_clip2f_nip (float *, const float *, float, float, long)

6.7.2.97 void dsp_clip2_nip (double *, const double *, double, double, long)

6.7.2.98 void dsp_clipzerof (float *, long)

[clDSPOp::ClipZero](#)

6.7.2.99 void dsp_clipzero (double *, long)

6.7.2.100 void dsp_clipzerof_nip (float *, const float *, long)

6.7.2.101 void dsp_clipzero_nip (double *, const double *, long)

6.7.2.102 void dsp_copyf (float *, const float *, long)

[clDSPOp::Copy](#)

6.7.2.103 void dsp_copy (double *, const double *, long)

6.7.2.104 float dsp_convolvef (const float *, const float *, long)

[clDSPOp::Convolve](#)

6.7.2.105 double dsp_convolve (const double *, const double *, long)

6.7.2.106 void dsp_convolve2f (float *, const float *, const float *, long)

6.7.2.107 void dsp_convolve2 (double *, const double *, const double *, long)

6.7.2.108 float dsp_correlatef (const float *, const float *, long)

[clDSPOp::Correlate](#)

6.7.2.109 **double dsp_correlate (const double *, const double *, long)**

6.7.2.110 **void dsp_correlate2f (float *, const float *, const float *, long)**

6.7.2.111 **void dsp_correlate2 (double *, const double *, const double *, long)**

6.7.2.112 **float dsp_autocorrf (const float *, long)**

[clDSPOp::AutoCorrelate](#)

6.7.2.113 **double dsp_autocorr (const double *, long)**

6.7.2.114 **void dsp_autocorr2f (float *, const float *, long)**

6.7.2.115 **void dsp_autocorr2 (double *, const double *, long)**

6.7.2.116 **float dsp_dotproductf (const float *, const float *, long)**

[clDSPOp::DotProduct](#)

6.7.2.117 **double dsp_dotproduct (const double *, const double *, long)**

6.7.2.118 **void dsp_minmaxf (float *, float *, const float *, long)**

[clDSPOp::MinMax](#)

6.7.2.119 **void dsp_minmax (double *, double *, const double *, long)**

6.7.2.120 **float dsp_meanf (const float *, long)**

[clDSPOp::Mean](#)

6.7.2.121 **double dsp_mean (const double *, long)**

6.7.2.122 **float dsp_meadianf (const float *, long)**

[clDSPOp::Median](#)

6.7.2.123 **double dsp_median (const double *, long)**

6.7.2.124 **void dsp_negatef (float *, long)**

[clDSPOp::Negate](#)

6.7.2.125 void dsp_negate (double *, long)

6.7.2.126 void dsp_negatef_nip (float *, const float *, long)

6.7.2.127 void dsp_negate_nip (double *, const double *, long)

6.7.2.128 void dsp_normalizef (float *, long)

[clDSPOp::Normalize](#)

6.7.2.129 void dsp_normalize (double *, long)

6.7.2.130 void dsp_normalizef_nip (float *, const float *, long)

6.7.2.131 void dsp_normalize_nip (double *, const double *, long)

6.7.2.132 float dsp_productf (const float *, long)

[clDSPOp::Product](#)

6.7.2.133 double dsp_product (const double *, long)

6.7.2.134 void dsp_reversef (float *, long)

[clDSPOp::Reverse](#)

6.7.2.135 void dsp_reverse (double *, long)

6.7.2.136 void dsp_reversef_nip (float *, const float *, long)

6.7.2.137 void dsp_reverse_nip (double *, const double *, long)

6.7.2.138 void dsp_scalef (float *, long)

[clDSPOp::Scale](#)

6.7.2.139 void dsp_scale (double *, long)

6.7.2.140 void dsp_scalef_nip (float *, const float *, long)

6.7.2.141 void dsp_scale_nip (double *, const double *, long)

6.7.2.142 void dsp_scale01f (float *, long)

[clDSPOp::Scale01](#)

6.7.2.143 void dsp_scale01 (double *, long)

6.7.2.144 void dsp_scale01f_nip (float *, const float *, long)

6.7.2.145 void dsp_Scale01_nip (double *, const double *, long)

6.7.2.146 void dsp_sortf (float *, long)

[clDSPOp::Sort](#)

6.7.2.147 void dsp_sort (double *, long)

6.7.2.148 void dsp_sortl (long *, long)

6.7.2.149 void dsp_stddevf (float *, float *, const float *, long)

[clDSPOp::StdDev](#)

6.7.2.150 void dsp_stddev (double *, double *, const double *, long)

6.7.2.151 float dsp_sumf (const float *, long)

[clDSPOp::Sum](#)

6.7.2.152 double dsp_sum (const double *, long)

6.7.2.153 void dsp_squaref (float *, long)

[clDSPOp::Square](#)

6.7.2.154 void dsp_square (double *, long)

6.7.2.155 void dsp_squaref_nip (float *, const float *, long)

6.7.2.156 void dsp_square_nip (double *, const double *, long)

6.7.2.157 void dsp_convertu8f (float *, const unsigned char *, long)

[clDSPOp::Convert](#)

- 6.7.2.158 void dsp_convertu8 (double *, const unsigned char *, long)
- 6.7.2.159 void dsp_converts16f (float *, const signed short *, long, int)
- 6.7.2.160 void dsp_converts16 (double *, const signed short *, long, int)
- 6.7.2.161 void dsp_converts32f (float *, const signed int *, long, int)
- 6.7.2.162 void dsp_converts32 (double *, const signed int *, long, int)
- 6.7.2.163 void dsp_converttd64f (float *, const double *, long)
- 6.7.2.164 void dsp_convertf32 (double *, const float *, long)
- 6.7.2.165 void dsp_convertf32c (unsigned char *, const float *, long)
- 6.7.2.166 void dsp_converttd64c (unsigned char *, const double *, long)
- 6.7.2.167 void dsp_convertf32s (signed short *, const float *, long, int)
- 6.7.2.168 void dsp_converttd64s (signed short *, const double *, long, int)
- 6.7.2.169 void dsp_convertf32i (signed int *, const float *, long, int)
- 6.7.2.170 void dsp_converttd64i (signed int *, const double *, long, int)
- 6.7.2.171 void dsp_cart2polarf (float *, float *, const float *, const float *, long)

[clDSPOp::CartToPolar](#)

- 6.7.2.172 void dsp_cart2polar (double *, double *, const double *, const double *, long)
- 6.7.2.173 void dsp_cart2polar2f (float *, float *, const *stpSCplx*, long)
- 6.7.2.174 void dsp_cart2polar2 (double *, double *, const *stpDCplx*, long)
- 6.7.2.175 void dsp_cart2polar3f ([stpSPolar](#), const *stpSCplx*, long)
- 6.7.2.176 void dsp_cart2polar3 ([stpDPolar](#), const *stpDCplx*, long)
- 6.7.2.177 void dsp_cart2polar4f ([utpSCoord](#), long)
- 6.7.2.178 void dsp_cart2polar4 ([utpDCoord](#), long)
- 6.7.2.179 void dsp_polar2cartf (float *, float *, const float *, const float *, long)

[clDSPOp::PolarToCart](#)

6.7.2.180 void dsp_polar2cart (double *, double *, const double *, const double *, long)

6.7.2.181 void dsp_polar2cart2f ([stpSCplx](#), const float *, const float *, long)

6.7.2.182 void dsp_polar2cart2 ([stpDCplx](#), const double *, const double *, long)

6.7.2.183 void dsp_polar2cart3f ([stpSCplx](#), const *stpSPolar*, long)

6.7.2.184 void dsp_polar2cart3 ([stpDCplx](#), const *stpDPolar*, long)

6.7.2.185 void dsp_polar2cart4f ([utpSCoord](#), long)

6.7.2.186 void dsp_polar2cart4 ([utpDCoord](#), long)

6.7.2.187 float dsp_crosscorrf (const float *, const float *, long)

[clDSPOp::CrossCorr](#)

6.7.2.188 double dsp_crosscorr (const double *, const double *, long)

6.7.2.189 float dsp_crosscorr2f (const float *, const float *, long, long)

[clDSPOp::DelCrossCorr](#)

6.7.2.190 double dsp_crosscorr2 (const double *, const double *, long, long)

6.7.2.191 void dsp_crosscorr3f (float *, const float *, const float *, long, const long *, long)

6.7.2.192 void dsp_crosscorr3 (double *, const double *, const double *, long, const long *, long)

6.7.2.193 float dsp_energyf (const float *, long)

[clDSPOp::Energy](#)

6.7.2.194 double dsp_energy (const double *, long)

6.7.2.195 void dsp_magnitudef (float *, const *stpSCplx*, long)

[clDSPOp::Magnitude](#)

6.7.2.196 void dsp_magnitude (double *, const *stpDCplx*, long)

6.7.2.197 void dsp_powerf (float *, const *stpSCplx*, long)

[clDSPOp::Power](#)

6.7.2.198 void dsp_power (double *, const *stpDCplx*, long)

6.7.2.199 void dsp_phasef (float *, const *stpSCplx*, long)

[clDSPOp::Phase](#)

6.7.2.200 void dsp_phase (double *, const *stpDCplx*, long)

6.7.2.201 void dsp_powerphasef (float *, float *, const *stpSCplx*, long)

[clDSPOp::PowerPhase](#)

6.7.2.202 void dsp_powerphase (double *, double *, const *stpDCplx*, long)

6.7.2.203 void dsp_decimatef (float *, const float *, long, long)

[clDSPOp::Decimate](#)

6.7.2.204 void dsp_decimate (double *, const double *, long, long)

6.7.2.205 void dsp_decimateavgf (float *, const float *, long, long)

[clDSPOp::DecimateAvg](#)

6.7.2.206 void dsp_decimateavg (double *, const double *, long, long)

6.7.2.207 void dsp_interpolatef (float *, const float *, long, long)

[clDSPOp::Interpolate](#)

6.7.2.208 void dsp_interpolate (double *, const double *, long, long)

6.7.2.209 void dsp_interpolateavgf (float *, const float *, long, long)

[clDSPOp::InterpolateAvg](#)

6.7.2.210 void dsp_interpolateavg (double *, const double *, long, long)

6.7.2.211 void dsp_resamplef (float *, long, const float *, long)

[clDSPOp::Resample](#)

6.7.2.212 void dsp_resample (double *, long, const double *, long)

6.7.2.213 void dsp_resampleavgf (float *, long, const float *, long)

[clDSPOp::ResampleAvg](#)

6.7.2.214 void dsp_resampleavg (double *, long, const double *, long)

6.7.2.215 float dsp_rmsf (const float *, long)

[clDSPOp::RMS](#)

6.7.2.216 double dsp_rms (const double *, long)

6.7.2.217 float dsp_variancef (float *, float *, const float *, long)

[clDSPOp::Variance](#)

6.7.2.218 double dsp_variance (double *, double *, const double *, long)

6.7.2.219 float dsp_peaklevelf (const float * *fpSrc*, long)

[clDSPOp::PeakLevel](#)

6.7.2.220 double dsp_peaklevel (const double * *dpSrc*, long)

6.7.2.221 void dsp_mixf (float *, const float *, long)

[clDSPOp::Mix](#)

6.7.2.222 void dsp_mix (double *, const double *, long)

6.7.2.223 void dsp_mix2f (float *, const float *, const float *, long)

6.7.2.224 void dsp_mix2 (double *, const double *, const double *, long)

6.7.2.225 void dsp_mixnf (float *, const float *, long, long)

6.7.2.226 void dsp_mixn (double *, const double *, long, long)

6.7.2.227 void dsp_extractf (float *, const float *, long, long, long)

[clDSPOp::Extract](#)

6.7.2.228 void dsp_extract (double *, const double *, long, long, long)

6.7.2.229 void dsp_packf (float *, const float *, long, long, long)

[clDSPOp::Pack](#)

6.7.2.230 void dsp_pack (double *, const double *, long, long, long)

6.7.2.231 void dsp_fftw_convertf2cf ([stpSCplx](#), const float *, long)

[clDSPOp::FFTWConvert](#)

6.7.2.232 void dsp_fftw_convertf2cd ([stpDCplx](#), const float *, long)

6.7.2.233 void dsp_fftw_convertf2d2cf ([stpSCplx](#), const double *, long)

6.7.2.234 void dsp_fftw_convertf2d2cd ([stpDCplx](#), const double *, long)

6.7.2.235 void dsp_fftw_convertf2cf2f (float *, const *stpSCplx*, long)

6.7.2.236 void dsp_fftw_convertf2cd2f (float *, const *stpDCplx*, long)

6.7.2.237 void dsp_fftw_convertf2cf2d (double *, const *stpSCplx*, long)

6.7.2.238 void dsp_fftw_convertf2cd2d (double *, const *stpDCplx*, long)

6.7.2.239 [dsp_t](#) dsp_new (void)

Creates new instance of DSP object.

Returns:

DSP object instance

6.7.2.240 void dsp_delete (dsp_t)

Deletes DSP object instance.

Parameters:

dspInst DSP object instance

6.7.2.241 void dsp_win_bartlett (dsp_t, float *, long)

[clDSPOp::WinBartlett](#)

6.7.2.242 void dsp_win_bartlett (dsp_t, double *, long)**6.7.2.243 void dsp_win_blackmanf (dsp_t, float *, long, float)**

[clDSPOp::WinBlackman](#)

6.7.2.244 void dsp_win_blackman (dsp_t, double *, long, double)**6.7.2.245 void dsp_win_blackman_harrisf (dsp_t, float *, long)**

[clDSPOp::WinBlackmanHarris](#)

6.7.2.246 void dsp_win_blackman_harris (dsp_t, double *, long)**6.7.2.247 void dsp_win_cos_taperedf (dsp_t, float *, long)**

[clDSPOp::WinCosTapered](#)

6.7.2.248 void dsp_win_cos_tapered (dsp_t, double *, long)**6.7.2.249 void dsp_win_exact_blackmanf (dsp_t, float *, long)**

[clDSPOp::WinExactBlackman](#)

6.7.2.250 void dsp_win_exact_blackman (dsp_t, double *, long)**6.7.2.251 void dsp_win_expf (dsp_t, float *, float, long)**

[clDSPOp::WinExp](#)

6.7.2.252 void dsp_win_exp ([dsp_t](#), double *, double, long)

6.7.2.253 void dsp_win_flat_topf ([dsp_t](#), float *, long)

[clDSPOp::WinFlatTop](#)

6.7.2.254 void dsp_win_flat_top ([dsp_t](#), double *, long)

6.7.2.255 void dsp_win_generic_cosf ([dsp_t](#), float *, long, const float *, long)

[clDSPOp::WinGenericCos](#)

6.7.2.256 void dsp_win_generic_cos ([dsp_t](#), double *, long, const double *, long)

6.7.2.257 void dsp_win_hammingf ([dsp_t](#), float *, long)

[clDSPOp::WinHamming](#)

6.7.2.258 void dsp_win_hamming ([dsp_t](#), double *, long)

6.7.2.259 void dsp_win_hanningf ([dsp_t](#), float *, long)

[clDSPOp::WinHanning](#)

6.7.2.260 void dsp_win_hanning ([dsp_t](#), double *, long)

6.7.2.261 void dsp_win_kaiserf ([dsp_t](#), float *, float, long)

[clDSPOp::WinKaiser](#)

6.7.2.262 void dsp_win_kaiser ([dsp_t](#), double *, double, long)

6.7.2.263 void dsp_win_kaiser_besself ([dsp_t](#), float *, float, long)

[clDSPOp::WinKaiserBessel](#)

6.7.2.264 void dsp_win_kaiser_bessel ([dsp_t](#), double *, double, long)

6.7.2.265 void dsp_win_tukeyf ([dsp_t](#), float *, long)

[clDSPOp::WinTukey](#)

6.7.2.266 void dsp_win_tukey ([dsp_t](#), double *, long)

6.7.2.267 void dsp_win_dolph_chebyshevf ([dsp_t](#), float *, float, long)

[clDSPOp::WinDolphChebyshev](#)

6.7.2.268 void dsp_win_dolph_chebyshev ([dsp_t](#), double *, double, long)

6.7.2.269 long dsp_rebufferf ([dsp_t](#), float *, const float *, long, long)

[clDSPOp::ReBuffer](#)

6.7.2.270 long dsp_rebuffer ([dsp_t](#), double *, const double *, long, long)

6.7.2.271 float dsp_deg2radf ([dsp_t](#), float)

[clDSPOp::DegToRad](#)

6.7.2.272 double dsp_deg2rad ([dsp_t](#), double)

6.7.2.273 float dsp_rad2degf ([dsp_t](#), float)

[clDSPOp::RadToDeg](#)

6.7.2.274 double dsp_rad2deg ([dsp_t](#), double)

6.7.2.275 void dsp_fir_allocatf ([dsp_t](#), const float *, long)

[clDSPOp::FIRAllocate](#)

6.7.2.276 void dsp_fir_allocate ([dsp_t](#), const double *, long)

6.7.2.277 void dsp_fir_free ([dsp_t](#))

[clDSPOp::FIRFree](#)

6.7.2.278 void dsp_fir_filterf ([dsp_t](#), float *, long)

[clDSPOp::FIRFilter](#)

6.7.2.279 void dsp_fir_filter ([dsp_t](#), double *, long)

6.7.2.280 void dsp_fir_filterf_nip ([dsp_t](#), float *, const float *, long)

6.7.2.281 void dsp_fir_filter_nip ([dsp_t](#), double *, const double *, long)

6.7.2.282 void dsp_fir_filterf_fst ([dsp_t](#), float *, float *, long)

[cIDSPOp::FIRFilterF](#)

6.7.2.283 void dsp_fir_filter_fst ([dsp_t](#), double *, double *, long)

6.7.2.284 void dsp_iir_initf ([dsp_t](#), const float *)

[cIDSPOp::IIRInitialize](#)

6.7.2.285 void dsp_iir_init ([dsp_t](#), const double *)

6.7.2.286 void dsp_iir_filterf ([dsp_t](#), float *, long)

[cIDSPOp::IIRFilter](#)

6.7.2.287 void dsp_iir_filter ([dsp_t](#), double *, long)

6.7.2.288 void dsp_iir_filterf_nip ([dsp_t](#), float *, const float *, long)

6.7.2.289 void dsp_iir_filter_nip ([dsp_t](#), double *, const double *, long)

6.7.2.290 void dsp_iir_clear ([dsp_t](#))

[cIDSPOp::IIRCLEAR](#)

6.7.2.291 void dsp_fft_init ([dsp_t](#), long, int)

[cIDSPOp::FFTInitialize](#)

6.7.2.292 void dsp_fft_uninit ([dsp_t](#))

[cIDSPOp::FFTUninitialize](#)

6.7.2.293 void dsp_fftf ([dsp_t](#), [stpSCplx](#), float *)

[cIDSPOp::FFTi](#)

6.7.2.294 void dsp_fft ([dsp_t](#), [stpDCplx](#), double *)

6.7.2.295 void dsp_fftf_nip ([dsp_t](#), [stpSCplx](#), const float *)

[clDSPOp::FFTo](#)

6.7.2.296 void dsp_fftf_nip ([dsp_t](#), [stpDCplx](#), const double *)

6.7.2.297 void dsp_cfftf_nip ([dsp_t](#), [stpSCplx](#), const *stpSCplx*)

6.7.2.298 void dsp_cfftf_nip ([dsp_t](#), [stpDCplx](#), const *stpDCplx*)

6.7.2.299 void dsp_ifftf_nip ([dsp_t](#), float *, const *stpSCplx*)

[clDSPOp::IFFTo](#)

6.7.2.300 void dsp_ifftf_nip ([dsp_t](#), double *, const *stpDCplx*)

6.7.2.301 void dsp_cifftf_nip ([dsp_t](#), [stpSCplx](#), const *stpSCplx*)

6.7.2.302 void dsp_cifftf_nip ([dsp_t](#), [stpDCplx](#), const *stpDCplx*)

6.7.2.303 [dsp_iircas_t](#) dsp_iir_cas_new (void)

Creates new instance of cascaded IIR object.

Returns:

Cascade IIR instance

6.7.2.304 void dsp_iir_cas_delete ([dsp_iircas_t](#))

Deletes cascaded IIR object instance.

Parameters:

dspiircasInst Cascaded IIR instance

6.7.2.305 int dsp_iir_cas_initf ([dsp_iircas_t](#), const *float*[][5], long)

[clIIRCascade::Initialize](#)

6.7.2.306 `int dsp_iir_cas_init (dsp_iircas_t, const double[][5], long)`

6.7.2.307 `void dsp_iir_cas_uninit (dsp_iircas_t)`

6.7.2.308 `void dsp_iir_cas_processf (dsp_iircas_t, float *, long)`

6.7.2.309 `void dsp_iir_cas_process (dsp_iircas_t, double *, long)`

6.7.2.310 `void dsp_iir_cas_processf_nip (dsp_iircas_t, float *, const float *, long)`

6.7.2.311 `void dsp_iir_cas_process_nip (dsp_iircas_t, double *, const double *, long)`

6.7.2.312 `void dsp_iir_cas_clear (dsp_iircas_t)`

6.7.2.313 `dsp_decfft_t dsp_dec_fft_new (void)`

Creates new instance of FFT decimator object.

Returns:

FFT decimator instance

6.7.2.314 `void dsp_dec_fft_delete (dsp_decfft_t)`

Deletes FFT decimator object instance.

Parameters:

dspdecfftInst FFT decimator instance

6.7.2.315 `int dsp_dec_fft_initf (dsp_decfft_t, long, long, int)`

[clFFTDecimator::Initialize](#)

6.7.2.316 `int dsp_dec_fft_init (dsp_decfft_t, long, long, int)`

6.7.2.317 `void dsp_dec_fft_uninit (dsp_decfft_t)`

[clFFTDecimator::Uninitialize](#)

6.7.2.318 `void dsp_dec_fft_putf (dsp_decfft_t, const float *, long)`

[clFFTDecimator::Put](#)

6.7.2.319 void dsp_dec_fft_put ([dsp_decfft_t](#), const double *, long)

6.7.2.320 int dsp_dec_fft_getf ([dsp_decfft_t](#), float *, long)

[clFFTDecimator::Get](#)

6.7.2.321 int dsp_dec_fft_get ([dsp_decfft_t](#), double *, long)

6.7.2.322 [dsp_decfir_t](#) dsp_dec_fir_new (void)

Creates new instance of FIR decimator object.

Returns:

FIR decimator instance

6.7.2.323 void dsp_dec_fir_delete ([dsp_decfir_t](#))

Deletes FIR decimator object instance.

Parameters:

dspdecfirInst FIR decimator instance

6.7.2.324 int dsp_dec_fir_initf ([dsp_decfir_t](#), long, int)

[clFIRDecimator::Initialize](#)

6.7.2.325 int dsp_dec_fir_init ([dsp_decfir_t](#), long, int)

6.7.2.326 void dsp_dec_fir_uninit ([dsp_decfir_t](#))

[clFIRDecimator::Uninitialize](#)

6.7.2.327 void dsp_dec_fir_putf ([dsp_decfir_t](#), const float *, long)

[clFIRDecimator::Put](#)

6.7.2.328 void dsp_dec_fir_put ([dsp_decfir_t](#), const double *, long)

6.7.2.329 int dsp_dec_fir_getf ([dsp_decfir_t](#), float *, long)

[clFIRDecimator::Get](#)

6.7.2.330 `int dsp_dec_fir_get (dsp_decfir_t, double *, long)`

6.7.2.331 `dsp_deciir_t dsp_dec_iir_new (void)`

Creates new instance of IIR decimator object.

Returns:

IIR decimator instance

6.7.2.332 `void dsp_dec_iir_delete (dsp_deciir_t)`

Deletes IIR decimator object instance.

Parameters:

dspdeciirInst IIR decimator instance

6.7.2.333 `int dsp_dec_iir_initf (dsp_deciir_t, long, int)`

[cIIRDecimator::Initialize](#)

6.7.2.334 `int dsp_dec_iir_init (dsp_deciir_t, long, int)`

6.7.2.335 `void dsp_dec_iir_uninit (dsp_deciir_t)`

[cIIRDecimator::Uninitialize](#)

6.7.2.336 `void dsp_dec_iir_putf (dsp_deciir_t, const float *, long)`

[cIIRDecimator::Put](#)

6.7.2.337 `void dsp_dec_iir_put (dsp_deciir_t, const double *, long)`

6.7.2.338 `int dsp_dec_iir_getf (dsp_deciir_t, float *, long)`

[cIIRDecimator::Get](#)

6.7.2.339 `int dsp_dec_iir_get (dsp_deciir_t, double *, long)`

6.7.2.340 `dsp_decrec_t dsp_dec_rec_new (void)`

Creates new instance of recursive decimator object.

Returns:

Recursive decimator instance

6.7.2.341 void `dsp_dec_rec_delete` ([dsp_decrec_t](#))

Deletes recursive decimator object instance.

Parameters:

dspdecrecInst Recursive decimator instance

6.7.2.342 int `dsp_dec_rec_initf` ([dsp_decrec_t](#), long, long, float, int)

[clRecDecimator::Initialize](#)

6.7.2.343 int `dsp_dec_rec_init` ([dsp_decrec_t](#), long, long, double, int)

6.7.2.344 void `dsp_dec_rec_uninit` ([dsp_decrec_t](#))

[clRecDecimator::Uninitialize](#)

6.7.2.345 void `dsp_dec_rec_putf` ([dsp_decrec_t](#), const float *, long)

[clRecDecimator::Put](#)

6.7.2.346 void `dsp_dec_rec_put` ([dsp_decrec_t](#), const double *, long)

6.7.2.347 int `dsp_dec_rec_getf` ([dsp_decrec_t](#), float *, long)

[clRecDecimator::Get](#)

6.7.2.348 int `dsp_dec_rec_get` ([dsp_decrec_t](#), double *, long)

6.7.2.349 [dsp_intfft_t](#) `dsp_int_fft_new` (void)

Creates new instance of FFT interpolator object.

Returns:

FFT interpolator instance

6.7.2.350 void `dsp_int_fft_delete` ([dsp_intfft_t](#))

Deletes FFT interpolator object instance.

Parameters:

dspintfftInst FFT interpolator instance

6.7.2.351 int dsp_int_fft_initf ([dsp_intfft_t](#), long, long, int)

[clFFTInterpolator::Initialize](#)

6.7.2.352 int dsp_int_fft_init ([dsp_intfft_t](#), long, long, int)

6.7.2.353 void dsp_int_fft_uninit ([dsp_intfft_t](#))

[clFFTInterpolator::Uninitialize](#)

6.7.2.354 void dsp_int_fft_putf ([dsp_intfft_t](#), const float *, long)

[clFFTInterpolator::Put](#)

6.7.2.355 void dsp_int_fft_put ([dsp_intfft_t](#), const double *, long)

6.7.2.356 int dsp_int_fft_getf ([dsp_intfft_t](#), float *, long)

[clFFTInterpolator::Get](#)

6.7.2.357 int dsp_int_fft_get ([dsp_intfft_t](#), double *, long)

6.7.2.358 [dsp_intfir_t](#) dsp_int_fir_new (void)

Creates new instance of FIR interpolator object.

Returns:

FIR interpolator instance

6.7.2.359 void dsp_int_fir_delete ([dsp_intfir_t](#))

Deletes FIR interpolator object instance.

Parameters:

dspintfirInst FIR interpolator instance

6.7.2.360 int dsp_int_fir_initf ([dsp_intfir_t](#), long, int)

[clFIRInterpolator::Initialize](#)

6.7.2.361 int dsp_int_fir_init ([dsp_intfir_t](#), long, int)

6.7.2.362 void dsp_int_fir_uninit ([dsp_intfir_t](#))

[clFIRInterpolator::Uninitialize](#)

6.7.2.363 void dsp_int_fir_putf ([dsp_intfir_t](#), const float *, long)

[clFIRInterpolator::Put](#)

6.7.2.364 void dsp_int_fir_put ([dsp_intfir_t](#), const double *, long)

6.7.2.365 int dsp_int_fir_getf ([dsp_intfir_t](#), float *, long)

[clFIRInterpolator::Get](#)

6.7.2.366 int dsp_int_fir_get ([dsp_intfir_t](#), double *, long)

6.7.2.367 [dsp_intiir_t](#) dsp_int_iir_new (void)

Creates new instance of IIR interpolator object.

Returns:

IIR interpolator instance

6.7.2.368 void dsp_int_iir_delete ([dsp_intiir_t](#))

Deletes IIR interpolator object instance.

Parameters:

[dspintiirInst](#) IIR interpolator instance

6.7.2.369 int dsp_int_iir_initf ([dsp_intiir_t](#), long, int)

[clIIRInterpolator::Initialize](#)

6.7.2.370 int dsp_int_iir_init ([dsp_intiir_t](#), long, int)

6.7.2.371 void dsp_int_iir_uninit ([dsp_intiir_t](#))

[clIIRInterpolator::Uninitialize](#)

6.7.2.372 void dsp_int_iir_putf ([dsp_intiir_t](#), const float *, long)

[clIIRInterpolator::Put](#)

6.7.2.373 void dsp_int_iir_put ([dsp_intiir_t](#), const double *, long)

6.7.2.374 int dsp_int_iir_getf ([dsp_intiir_t](#), float *, long)

[clIIRInterpolator::Get](#)

6.7.2.375 int dsp_int_iir_get ([dsp_intiir_t](#), double *, long)

6.7.2.376 [dsp_intrec_t](#) dsp_int_rec_new (void)

Creates new instance of recursive interpolator object.

Returns:

Recursive interpolator instance

6.7.2.377 void dsp_int_rec_delete ([dsp_intrec_t](#))

Deletes recursive interpolator object instance.

Parameters:

dspdecrecInst Recursive interpolator instance

6.7.2.378 int dsp_int_rec_initf ([dsp_intrec_t](#), long, long, float, int)

[clRecInterpolator::Initialize](#)

6.7.2.379 int dsp_int_rec_init ([dsp_intrec_t](#), long, long, double, int)

6.7.2.380 void dsp_int_rec_uninit ([dsp_intrec_t](#))

[clRecInterpolator::Uninitialize](#)

6.7.2.381 void dsp_int_rec_putf ([dsp_intrec_t](#), const float *, long)

[clRecInterpolator::Put](#)

6.7.2.382 void dsp_int_rec_put ([dsp_intrec_t](#), const double *, long)

6.7.2.383 int dsp_int_rec_getf ([dsp_intrec_t](#), float *, long)

[clRecInterpolator::Get](#)

6.7.2.384 int dsp_int_rec_get ([dsp_intrec_t](#), double *, long)

6.7.2.385 [dsp_filter_t](#) dsp_filter_new (void)

Creates new instance of FFT filter object.

Returns:

FFT filter instance

6.7.2.386 void dsp_filter_delete ([dsp_filter_t](#))

Deletes instance of FFT filter object.

Parameters:

dspfilterInst FFT filter instance

6.7.2.387 int dsp_filter_initf ([dsp_filter_t](#), long)

[clFilter::Initialize](#)

6.7.2.388 int dsp_filter_init ([dsp_filter_t](#), long)

6.7.2.389 int dsp_filter_init2f ([dsp_filter_t](#), long, const float *, float, float)

6.7.2.390 int dsp_filter_init2 ([dsp_filter_t](#), long, const double *, double, double)

6.7.2.391 int dsp_filter_init_lpf ([dsp_filter_t](#), float, float, float, float)

[clFilter::InitializeLP](#)

6.7.2.392 int dsp_filter_init_lp ([dsp_filter_t](#), double, double, double, double)

6.7.2.393 int dsp_filter_init_hpf ([dsp_filter_t](#), float, float, float, float)

[clFilter::InitializeHP](#)

6.7.2.394 int dsp_filter_init_hp ([dsp_filter_t](#), double, double, double, double)

6.7.2.395 void dsp_filter_set_coefssf ([dsp_filter_t](#), const float *)

[clFilter::SetCoeffs](#)

6.7.2.396 void dsp_filter_set_ceoffs ([dsp_filter_t](#), const double *)

6.7.2.397 void dsp_filter_set_ccoeffssf ([dsp_filter_t](#), const *stpSCplx*, int)

6.7.2.398 void dsp_filter_set_ccoeffs ([dsp_filter_t](#), const *stpDCplx*, int)

6.7.2.399 void dsp_filter_get_coefssf ([dsp_filter_t](#), float *)

[clFilter::GetCoeffs](#)

6.7.2.400 void dsp_filter_get_ceoffs ([dsp_filter_t](#), double *)

6.7.2.401 void dsp_filter_get_ccoeffssf ([dsp_filter_t](#), *stpSCplx*)

6.7.2.402 void dsp_filter_get_ccoeffs ([dsp_filter_t](#), *stpDCplx*)

6.7.2.403 void dsp_filter_putf ([dsp_filter_t](#), const float *, long)

[clFilter::Put](#)

6.7.2.404 void dsp_filter_put ([dsp_filter_t](#), const double *, long)

6.7.2.405 void dsp_filter_put2f ([dsp_filter_t](#), const float *, long, const *stpSCplx*)

6.7.2.406 void dsp_filter_put2 ([dsp_filter_t](#), const double *, long, const *stpDCplx*)

6.7.2.407 void dsp_filter_getf ([dsp_filter_t](#), float *, long)

[clFilter::Get](#)

6.7.2.408 void dsp_filter_get ([dsp_filter_t](#), double *, long)

6.7.2.409 void dsp_filter_design_lpf ([dsp_filter_t](#), float *, int)

[clFilter::DesignLP](#)

6.7.2.410 void dsp_filter_design_lp ([dsp_filter_t](#), double *, int)

6.7.2.411 void dsp_filter_design_lp2f ([dsp_filter_t](#), float *, float, int)

6.7.2.412 void dsp_filter_design_lp2 ([dsp_filter_t](#), double *, double, int)

6.7.2.413 void dsp_filter_design_hpf ([dsp_filter_t](#), float *)

[clFilter::DesignHP](#)

6.7.2.414 void dsp_filter_design_hp ([dsp_filter_t](#), double *)

6.7.2.415 void dsp_filter_design_hp2f ([dsp_filter_t](#), float *, float)

6.7.2.416 void dsp_filter_design_hp2 ([dsp_filter_t](#), double *, double)

6.7.2.417 void dsp_filter_design_bpf ([dsp_filter_t](#), float *, float *)

[clFilter::DesignBP](#)

6.7.2.418 void dsp_filter_design_bp ([dsp_filter_t](#), double *, double *)

6.7.2.419 void dsp_filter_design_bp2f ([dsp_filter_t](#), float *, float *, float)

6.7.2.420 void dsp_filter_design_bp2 ([dsp_filter_t](#), double *, double *, double)

6.7.2.421 void dsp_filter_design_brf ([dsp_filter_t](#), float *, float *)

[clFilter::DesignBR](#)

6.7.2.422 void dsp_filter_design_br ([dsp_filter_t](#), double *, double *)

6.7.2.423 void dsp_filter_design_br2f ([dsp_filter_t](#), float *, float *, float)

6.7.2.424 void dsp_filter_design_br2 ([dsp_filter_t](#), double *, double *, double)

6.7.2.425 [dsp_rebuf_t](#) dsp_rebuf_new (void)

Creates new instance of rebuffering object.

Returns:

Rebuffer instance

6.7.2.426 void dsp_rebuf_delete ([dsp_rebuf_t](#))

Deletes instance of rebuffering object.

Parameters:

dsprebufInst Rebuffer instance

6.7.2.427 void dsp_rebuf_putf ([dsp_rebuf_t](#), const float *, long)

[clReBuffer::Put](#)

6.7.2.428 void dsp_rebuf_put ([dsp_rebuf_t](#), const double *, long)

6.7.2.429 int dsp_rebuf_getf ([dsp_rebuf_t](#), float *, long)

[clReBuffer::Get](#)

6.7.2.430 int dsp_rebuf_get ([dsp_rebuf_t](#), double *, long)

6.7.2.431 long dsp_rebuf_size ([dsp_rebuf_t](#))

[clReBuffer::GetCount](#)

6.7.2.432 void dsp_rebuf_clear ([dsp_rebuf_t](#))

[clReBuffer::Clear](#)

6.7.2.433 void dsp_rebuf_copy ([dsp_rebuf_t](#), [dsp_rebuf_t](#))

[clReBuffer::operator=](#)

6.8 DSPOp.hh File Reference

```
#include <math.h>
#include <float.h>
#include <Alloc.hh>
#include "dsp/dsptypes.h"
#include "dsp/DSPConfig.hh"
#include "dsp/Transform4.hh"
#include "dsp/Transform8.hh"
#include "dsp/TransformS.hh"
```

Classes

- class [cIDSPAlloc](#)
Class specialization to support automatic typecasts to cartesian/polar datatypes.
- class [cDSPOp](#)
Class of DSP operations.

Defines

- #define [CONSTFUNC](#)
- #define [INLINE](#) inline
- #define [DSP_MAXBESSEL](#) 32L
- #define [DSP_WISDOM_FILE](#) "fftw.wisdom"

6.8.1 Define Documentation

6.8.1.1 #define CONSTFUNC

6.8.1.2 #define INLINE inline

6.8.1.3 #define DSP_MAXBESSEL 32L

6.8.1.4 #define DSP_WISDOM_FILE "fftw.wisdom"

6.9 dsptypes.h File Reference

Classes

- struct [_sSCplx](#)
Single precision complex datatype.
- struct [_sDCplx](#)
Double precision complex datatype.
- struct [_sSPolar](#)
Single precision polar datatype.
- struct [_sDPolar](#)
Double precision polar datatype.
- union [_uSCoord](#)
Single precision cartesian/polar datatype.
- union [_uDCoord](#)
Double precision cartesian/polar datatype.

Typedefs

- typedef [_sSCplx](#) [stSCplx](#)
Single precision complex datatype.
- typedef [_sSCplx](#) * [stpSCplx](#)
Single precision complex datatype.
- typedef [_sDCplx](#) [stDCplx](#)
Double precision complex datatype.
- typedef [_sDCplx](#) * [stpDCplx](#)
Double precision complex datatype.
- typedef [_sSPolar](#) [stSPolar](#)
Single precision polar datatype.
- typedef [_sSPolar](#) * [stpSPolar](#)
Single precision polar datatype.
- typedef [_sDPolar](#) [stDPolar](#)
Double precision polar datatype.

- typedef [_sDPolar](#) * [stpDPolar](#)
Double precision polar datatype.
- typedef [_uSCoord](#) [utSCoord](#)
Single precision cartesian/polar datatype.
- typedef [_uSCoord](#) * [utpSCoord](#)
Single precision cartesian/polar datatype.
- typedef [_uDCoord](#) [utDCoord](#)
Double precision cartesian/polar datatype.
- typedef [_uDCoord](#) * [utpDCoord](#)
Double precision cartesian/polar datatype.

6.9.1 Typedef Documentation

6.9.1.1 typedef struct [_sSCplx](#) [stSCplx](#)

Single precision complex datatype.

Type prefix is left out to prevent need of source code changes when precision is changed.

6.9.1.2 typedef struct [_sSCplx](#) * [stpSCplx](#)

Single precision complex datatype.

Type prefix is left out to prevent need of source code changes when precision is changed.

6.9.1.3 typedef struct [_sDCplx](#) [stDCplx](#)

Double precision complex datatype.

6.9.1.4 typedef struct [_sDCplx](#) * [stpDCplx](#)

Double precision complex datatype.

6.9.1.5 typedef struct [_sSPolar](#) [stSPolar](#)

Single precision polar datatype.

6.9.1.6 typedef struct [_sSPolar](#) * [stpSPolar](#)

Single precision polar datatype.

6.9.1.7 typedef struct [_sDPolar](#) [stDPolar](#)

Double precision polar datatype.

6.9.1.8 typedef struct [_sDPolar](#) * [stpDPolar](#)

Double precision polar datatype.

6.9.1.9 typedef union [_uSCoord](#) [utSCoord](#)

Single precision cartesian/polar datatype.

6.9.1.10 typedef union [_uSCoord](#) * [utpSCoord](#)

Single precision cartesian/polar datatype.

6.9.1.11 typedef union [_uDCoord](#) [utDCoord](#)

Double precision cartesian/polar datatype.

6.9.1.12 typedef union [_uDCoord](#) * [utpDCoord](#)

Double precision cartesian/polar datatype.

6.10 DSPVector.hh File Reference

```
#include "dsp/DSPOp.hh"  
#include "dsp/ReBufferT.hh"
```

Classes

- class [clDSPVector](#)

Defines

- #define [DSPV_SHORTER](#)(x) ((Size() <= x.Size()) ? Size() : x.Size())
- #define [DSPV_SHORTER2](#)(x, y) ((x.Size() <= y.Size()) ? x.Size() : y.Size())

6.10.1 Define Documentation

6.10.1.1 #define [DSPV_SHORTER](#)(x) ((Size() <= x.Size()) ? Size() : x.Size())

6.10.1.2 #define [DSPV_SHORTER2](#)(x, y) ((x.Size() <= y.Size()) ? x.Size() : y.Size())

6.11 DynThreads.cc File Reference

```
#include <cstdio>
#include <climits>
#include <pthread.h>
#include "DynThreads.hh"
```

Functions

- void * [WrapDynThreadBase](#) (void *vpParam)

6.11.1 Function Documentation

6.11.1.1 void* `WrapDynThreadBase` (void * *vpParam*)

6.12 DynThreads.hh File Reference

```
#include <unistd.h>
#include <sys/types.h>
#include <pthread.h>
#include <map>
#include <Mutex.hh>
```

Classes

- class [clDynThreadsBase](#)
- struct [clDynThreadsBase::_stParams](#)
- class [clDynThreads](#)
- struct [clDynThreads::_stParams2](#)

Typedefs

- typedef std::map< int, pthread_t > [ThreadMap_t](#)

6.12.1 Typedef Documentation

6.12.1.1 typedef std::map<int, pthread_t> [ThreadMap_t](#)

6.13 Exception.hh File Reference

```
#include <exception>
#include <string>
#include <cstring>
#include <cstdio>
```

Classes

- class [clException](#)

6.14 FFTDecimator.cc File Reference

```
#include <stdio.h>
#include <math.h>
#include <float.h>
#include "dsp/FFTDecimator.hh"
```

6.15 FFTDecimator.hh File Reference

```
#include "dsp/DSPOp.hh"  
#include "dsp/FFTMultiRate.hh"
```

Classes

- class [clFFTDecimator](#)
FFT decimation filter class implementation.

6.16 FFTInterpolator.cc File Reference

```
#include <stdio.h>
#include <math.h>
#include <float.h>
#include "dsp/FFTInterpolator.hh"
```

6.17 FFTInterpolator.hh File Reference

```
#include "dsp/DSPOp.hh"  
#include "dsp/FFTMultiRate.hh"
```

Classes

- class [clFFTInterpolator](#)
FFT interpolation filter class implementation.

6.18 FFTMultiRate.cc File Reference

```
#include <stdio.h>
#include <math.h>
#include <float.h>
#include "dsp/FFTMultiRate.hh"
```

6.19 FFTMultiRate.hh File Reference

```
#include "dsp/Filter.hh"
```

Classes

- class [clFFTMultiRate](#)
Base class for FFT based multirate filters.

Defines

- #define [FFTMULTIRATE_RIPPLERATIO](#) 144.49f
- #define [FFTMULTIRATE_OVERLAP](#) 0.75f
- #define [FFTMULTIRATE_DELTAOMEGA](#) 0.1f

6.19.1 Define Documentation

6.19.1.1 #define [FFTMULTIRATE_RIPPLERATIO](#) 144.49f

6.19.1.2 #define [FFTMULTIRATE_OVERLAP](#) 0.75f

6.19.1.3 #define [FFTMULTIRATE_DELTAOMEGA](#) 0.1f

6.20 Filter.cc File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <float.h>
#include "dsp/Filter.hh"
```

6.21 Filter.hh File Reference

```
#include "dsp/DSPConfig.hh"
#include "dsp/DSPOp.hh"
#include "dsp/ReBuffer.hh"
```

Classes

- class [clFilter](#)
Class implementing FFT-based FIR-filter with design functions.

Enumerations

- enum { [FILTER_SMOOTH_NONE](#) = 0, [FILTER_SMOOTH_KAISER](#) = 1, [FILTER_SMOOTH_KAISER_BESSEL](#) = 2, [FILTER_SMOOTH_DOLPH-CHEBYSHEV](#) = 3 }
Filter smoothing window types.

6.21.1 Enumeration Type Documentation

6.21.1.1 anonymous enum

Filter smoothing window types.

Enumeration values:

FILTER_SMOOTH_NONE
FILTER_SMOOTH_KAISER
FILTER_SMOOTH_KAISER_BESSEL
FILTER_SMOOTH_DOLPH-CHEBYSHEV

6.22 FIRDecimator.cc File Reference

```
#include <stdio.h>
#include <math.h>
#include <float.h>
#include "dsp/FIRDecimator.hh"
```

6.23 FIRDecimator.hh File Reference

```
#include <Alloc.hh>
#include "dsp/DSPConfig.hh"
#include "dsp/DSPOp.hh"
#include "dsp/ReBuffer.hh"
#include "dsp/FIRMultiRate.hh"
```

Classes

- class [clFIRDecimator](#)
FIR decimation filter class implementation.

6.24 FIRInterpolator.cc File Reference

```
#include <stdio.h>
#include <math.h>
#include <float.h>
#include "dsp/FIRInterpolator.hh"
```

6.25 FIRInterpolator.hh File Reference

```
#include "dsp/DSPConfig.hh"
#include "dsp/DSPOp.hh"
#include "dsp/ReBuffer.hh"
#include "dsp/FIRMultiRate.hh"
```

Classes

- class [cFIRInterpolator](#)
FIR interpolation filter class implementation.

6.26 FIRMultiRate.cc File Reference

```
#include <stdio.h>
#include <math.h>
#include <float.h>
#include "dsp/FIRMultiRate.hh"
#include "Dec2Filter3.h"
#include "Dec3Filter3.h"
#include "Dec4Filter3.h"
#include "Dec8Filter3.h"
#include "Dec2hpFilter3.h"
#include "Dec3hpFilter3.h"
#include "Dec4hpFilter3.h"
#include "Dec8hpFilter3.h"
```

6.27 FIRMultiRate.hh File Reference

```
#include "dsp/DSPOp.hh"
```

Classes

- class [clFIRMultiRate](#)
Base class for FIR based multirate filters.

6.28 FlipBand.cc File Reference

```
#include "dsp/FlipBand.hh"
```

6.29 FlipBand.hh File Reference

```
#include "DSPOp.hh"  
#include "ReBuffer.hh"
```

Classes

- class [clFlipBand](#)

6.30 Hankel.cc File Reference

```
#include <math.h>
#include <float.h>
#include "dsp/Hankel.hh"
```

Defines

- #define [ABEL_NSE](#) 9

Variables

- const float [fpH](#) [ABEL_NSE]
- const double [dpH](#) [ABEL_NSE]
- const float [fpLambda](#) [ABEL_NSE]
- const double [dpLambda](#) [ABEL_NSE]

6.30.1 Define Documentation

6.30.1.1 #define ABEL_NSE 9

6.30.2 Variable Documentation

6.30.2.1 const float [fpH](#)[ABEL_NSE]

Initial value:

```
{
  1.0000000000000000f,
  0.610926299405048390f,
  0.895089852938535935f,
  1.34082948787002865f,
  2.02532848558443890f,
  3.18110895533701843f,
  5.90898360396353794f,
  77.6000213494180286f,
  528.221800846070892f
}
```

6.30.2.2 const double [dpH](#)[ABEL_NSE]

Initial value:

```
{
  1.0000000000000000,
  0.610926299405048390,
  0.895089852938535935,
```

```
1.34082948787002865,  
2.02532848558443890,  
3.18110895533701843,  
5.90898360396353794,  
77.6000213494180286,  
528.221800846070892  
}
```

6.30.2.3 const float [fpLambda](#)[ABEL_NSE]

Initial value:

```
{  
  0.0000000000000000f,  
  -2.08424632126539366f,  
  -5.78928630565552371f,  
  -14.6268676854951032f,  
  -35.0617158334443104f,  
  -83.3258406398958158f,  
  -210.358805421311445f,  
  -6673.64911325382036f,  
  -34897.7050244132261f  
}
```

6.30.2.4 const double [dpLambda](#)[ABEL_NSE]

Initial value:

```
{  
  0.0000000000000000,  
  -2.08424632126539366,  
  -5.78928630565552371,  
  -14.6268676854951032,  
  -35.0617158334443104,  
  -83.3258406398958158,  
  -210.358805421311445,  
  -6673.64911325382036,  
  -34897.7050244132261  
}
```

6.31 Hankel.hh File Reference

```
#include <dsp/DSPOp.hh>
```

Classes

- class [clHankel](#)
Class implementation of (modified) Hankel-transform.

6.32 IIRCascade.cc File Reference

```
#include "dsp/IIRCascade.hh"
```


6.33 IIRCascade.hh File Reference

```
#include "dsp/DSPOp.hh"
```

Classes

- class [cIIRCascade](#)
Class to handle cascaded IIR stages as one filter.

6.34 IIRDecimator.cc File Reference

```
#include <stdio.h>
#include <math.h>
#include <float.h>
#include "dsp/IIRDecimator.hh"
```

6.35 IIRDecimator.hh File Reference

```
#include "dsp/DSPConfig.hh"  
#include "dsp/DSPOp.hh"  
#include "dsp/ReBuffer.hh"  
#include "dsp/IIRMultiRate.hh"
```

Classes

- class [cIIRDecimator](#)
IIR decimation filter class implementation.

6.36 IIRInterpolator.cc File Reference

```
#include <stdio.h>
#include <math.h>
#include <float.h>
#include "dsp/IIRInterpolator.hh"
```

6.37 IIRInterpolator.hh File Reference

```
#include "dsp/DSPConfig.hh"  
#include "dsp/DSPOp.hh"  
#include "dsp/ReBuffer.hh"  
#include "dsp/IIRMultiRate.hh"
```

Classes

- class [cIIRInterpolator](#)
IIR interpolation filter class implementation.

6.38 IIRMultiRate.cc File Reference

```
#include <stdio.h>
#include <math.h>
#include <float.h>
#include "dsp/IIRMultiRate.hh"
#include "Dec2IIR3.h"
#include "Dec2hpIIR3.h"
#include "Dec3IIR3.h"
#include "Dec3hpIIR3.h"
```

6.39 IIRMultiRate.hh File Reference

```
#include "dsp/DSPOp.hh"  
#include "dsp/IIRCascade.hh"
```

Classes

- class [cIIRMultiRate](#)
Base class for IIR based multirate filters.

6.40 Mutex.hh File Reference

```
#include <pthread.h>
```

Classes

- class [clMutex](#)
Class implementation of POSIX mutex semaphore.

6.41 PthCond.hh File Reference

```
#include <pth.h>
```

Classes

- class [clPthCond](#)

6.42 PthMutex.hh File Reference

```
#include <pth.h>
```

Classes

- class [clPthMutex](#)

6.43 ReBuffer.cc File Reference

```
#include <math.h>
#include <float.h>
#include <string.h>
#include <stdio.h>
#include <Exception.hh>
#include "dsp/ReBuffer.hh"
```

6.44 ReBuffer.hh File Reference

```
#include <typeinfo>
#include <Alloc.hh>
```

Classes

- class [clReBuffer](#)
Class for splitting data into buffers of different sizes.

6.45 ReBufferT.hh File Reference

```
#include <cmath>
#include <cfloat>
#include <cstring>
#include <typeinfo>
#include <Exception.hh>
#include "dsp/DSPOp.hh"
```

Classes

- class [clReBufferT](#)

Template class for splitting data into buffers of different sizes.

6.46 RecDecimator.cc File Reference

```
#include <stdio.h>
#include <math.h>
#include <float.h>
#include "dsp/RecDecimator.hh"
```

6.47 RecDecimator.hh File Reference

```
#include "dsp/DSPConfig.hh"
#include "dsp/DSPOp.hh"
#include "dsp/FFTDecimator.hh"
#include "dsp/FIRDecimator.hh"
#include "dsp/IIRDecimator.hh"
```

Classes

- class [clRecDecimator](#)
Recursive decimation filter class implementation.

Defines

- #define [RECDEC_MAX_SUB_ROUNDS](#) 32
Maximum number of subsequent rounds.

6.47.1 Define Documentation

6.47.1.1 #define RECDEC_MAX_SUB_ROUNDS 32

Maximum number of subsequent rounds.

6.48 RecInterpolator.cc File Reference

```
#include <stdio.h>
#include <math.h>
#include <float.h>
#include "dsp/RecInterpolator.hh"
```


6.49 RecInterpolator.hh File Reference

```
#include "dsp/DSPConfig.hh"
#include "dsp/DSPOp.hh"
#include "dsp/FFTInterpolator.hh"
#include "dsp/FIRInterpolator.hh"
#include "dsp/IIRInterpolator.hh"
```

Classes

- class [clRecInterpolator](#)
Recursive interpolation filter class implementation.

Defines

- #define [RECINT_MAX_SUB_ROUNDS](#) 32
Maximum number of subsequent rounds.

6.49.1 Define Documentation

6.49.1.1 #define RECINT_MAX_SUB_ROUNDS 32

Maximum number of subsequent rounds.

6.50 RWLock.hh File Reference

```
#include <pthread.h>
```

Classes

- class [clRWLock](#)
Class implementation of SUSv2 read-write locks.

Defines

- `#define _XOPEN_SOURCE 600`

6.50.1 Define Documentation

6.50.1.1 `#define _XOPEN_SOURCE 600`

6.51 Semaphore.hh File Reference

```
#include <semaphore.h>
```

Classes

- class [clSemaphore](#)
Class implementation of POSIX counting semaphores.

6.52 Test.cc File Reference

```
#include <cstdio>
#include <unistd.h>
#include "DynThreads.hh"
```

Classes

- class [clUserThreads](#)

Functions

- int [main](#) (int argc, char *argv[])

6.52.1 Function Documentation

6.52.1.1 int main (int *argc*, char * *argv*[])

6.53 Transform4.cc File Reference

```
#include <math.h>
#include <float.h>
#include "dsp/Transform4.hh"
```

6.54 Transform4.hh File Reference

Classes

- class [cTransform4](#)
Decimation-in-frequency radix-2/4 transform.

Defines

- #define [T4_INLINE](#)

6.54.1 Define Documentation

6.54.1.1 #define T4_INLINE

6.55 Transform8.cc File Reference

```
#include <math.h>
#include <float.h>
#include "dsp/Transform8.hh"
```

6.56 Transform8.hh File Reference

Classes

- class [clTransform8](#)
Decimation-in-frequency radix-2/4/8 transform.

Defines

- #define [T8_INLINE](#)

6.56.1 Define Documentation

6.56.1.1 #define T8_INLINE

6.57 TransformS.cc File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <float.h>
#include "dsp/TransformS.hh"
```

6.58 TransformS.hh File Reference

```
#include "dsp/DSPConfig.hh"
```

Classes

- class [clTransformS](#)
Decimation-in-frequency split-radix transform.

Defines

- #define [TS_INLINE](#)

6.58.1 Define Documentation

6.58.1.1 #define TS_INLINE

6.59 X86-64.c File Reference

6.60 X86-64.h File Reference

6.61 X86.c File Reference

6.62 X86.h File Reference

Index

- ~clAlloc
 - clAlloc, [17](#)
- ~clCondition
 - clCondition, [22](#)
- ~clDSPOp
 - clDSPOp, [40](#)
- ~clDSPVector
 - clDSPVector, [115](#)
- ~clDynThreadsBase
 - clDynThreadsBase, [122](#)
- ~clException
 - clException, [125](#)
- ~clFFTDecimator
 - clFFTDecimator, [126](#)
- ~clFFTInterpolator
 - clFFTInterpolator, [128](#)
- ~clFFTMultiRate
 - clFFTMultiRate, [131](#)
- ~clFIRDecimator
 - clFIRDecimator, [144](#)
- ~clFIRInterpolator
 - clFIRInterpolator, [146](#)
- ~clFIRMultiRate
 - clFIRMultiRate, [149](#)
- ~clFilter
 - clFilter, [135](#)
- ~clFlipBand
 - clFlipBand, [152](#)
- ~clHankel
 - clHankel, [155](#)
- ~clIIRCascade
 - clIIRCascade, [160](#)
- ~clIIRDecimator
 - clIIRDecimator, [162](#)
- ~clIIRInterpolator
 - clIIRInterpolator, [164](#)
- ~clIIRMultiRate
 - clIIRMultiRate, [166](#)
- ~clMutex
 - clMutex, [168](#)
- ~clPthCond
 - clPthCond, [170](#)
- ~clPthMutex
 - clPthMutex, [171](#)
- ~clRWLock
 - clRWLock, [190](#)
- ~clReBuffer
 - clReBuffer, [173](#)
- ~clReBufferT
 - clReBufferT, [177](#)
- ~clRecDecimator
 - clRecDecimator, [181](#)
- ~clRecInterpolator
 - clRecInterpolator, [186](#)
- ~clSemaphore
 - clSemaphore, [194](#)
- ~clTransformS
 - clTransformS, [212](#)
- _XOPEN_SOURCE
 - RWLock.hh, [372](#)
- _sDCplx, [9](#)
- ~sDCplx
 - I, [9](#)
 - R, [9](#)
- _sDPolar, [10](#)
- ~sDPolar
 - M, [10](#)
 - P, [10](#)
- _sSCplx, [11](#)
- ~sSCplx
 - I, [11](#)
 - R, [11](#)
- _sSPolar, [12](#)
- ~sSPolar
 - M, [12](#)
 - P, [12](#)
- _uDCoord, [13](#)
- ~uDCoord
 - C, [13](#)
 - P, [13](#)
- _uSCoord, [14](#)
- ~uSCoord

- C, [14](#)
- P, [14](#)
- A
 - clHankel, [158](#)
- ABEL_NSE
 - Hankel.cc, [351](#)
- Abs
 - clDSPOp, [62](#)
 - clDSPVector, [115](#)
- Add
 - clDSPOp, [50–52](#)
- Alloc.hh, [215](#)
 - ALLOC_ALIGNMENT, [215](#)
 - ALLOC_USE_ALIGN, [215](#)
- ALLOC_ALIGNMENT
 - Alloc.hh, [215](#)
- ALLOC_USE_ALIGN
 - Alloc.hh, [215](#)
- AutoCorr
 - clDSPVector, [115](#)
- AutoCorrelate
 - clDSPOp, [71](#)
- B0
 - clHankel, [158](#)
- B1
 - clHankel, [158](#)
- bFFTInitialized
 - clDSPOp, [110](#)
- bInitialized
 - clFFTMultiRate, [131](#)
 - clFilter, [143](#)
 - clFlipBand, [153](#)
 - clIIRCascade, [161](#)
 - clRecDecimator, [183](#)
 - clRecInterpolator, [188](#)
- bitrv2
 - clTransform4, [199](#)
 - clTransform8, [208](#)
 - clTransformS, [212](#)
- bitrv208
 - clTransformS, [212](#)
- bitrv208neg
 - clTransformS, [212](#)
- bitrv216
 - clTransformS, [212](#)
- bitrv216neg
 - clTransformS, [212](#)
- bitrv2conj
 - clTransform4, [199](#)
 - clTransform8, [208](#)
 - clTransformS, [212](#)
- bLocked
 - clAlloc, [21](#)
- bpHalves
 - clRecDecimator, [183](#)
 - clRecInterpolator, [189](#)
- bRealTransform
 - clDSPOp, [110](#)
- bRun
 - clUserThreads, [213](#)
- Buffer
 - clReBuffer, [174](#)
 - clReBufferT, [179](#)
- C
 - _uDCoord, [13](#)
 - _uSCoord, [14](#)
- Cart2Polar
 - clDSPOp, [40, 41](#)
- CartToPolar
 - clDSPOp, [83, 84](#)
 - clDSPVector, [115](#)
- CCoeffs
 - clFilter, [143](#)
- cdft
 - clTransform4, [199, 200](#)
 - clTransform8, [208](#)
 - clTransformS, [212](#)
- cdspop.cc, [216](#)
 - dsp_abs, [242](#)
 - dsp_abs_nip, [242](#)
 - dsp_absf, [242](#)
 - dsp_absf_nip, [242](#)
 - dsp_add, [236](#)
 - dsp_add2, [237](#)
 - dsp_add2f, [237](#)
 - dsp_add3, [237](#)
 - dsp_add3f, [237](#)
 - dsp_addf, [236](#)
 - dsp_autocorr, [245](#)
 - dsp_autocorr2, [245](#)
 - dsp_autocorr2f, [245](#)
 - dsp_autocorrf, [245](#)
 - dsp_cadd, [237](#)
 - dsp_cadd2, [237](#)
 - dsp_cadd2f, [237](#)
 - dsp_cadd3, [237](#)
 - dsp_cadd3f, [237](#)

[dsp_caddf](#), 237
[dsp_cart2polar](#), 249
[dsp_cart2polar2](#), 250
[dsp_cart2polar2f](#), 250
[dsp_cart2polar3](#), 250
[dsp_cart2polar3f](#), 250
[dsp_cart2polar4](#), 250
[dsp_cart2polar4f](#), 250
[dsp_cart2polarf](#), 249
[dsp_ccmul](#), 240
[dsp_ccmul_nip](#), 240
[dsp_ccmulf](#), 240
[dsp_ccmulf_nip](#), 240
[dsp_ccopy](#), 244
[dsp_ccopyf](#), 244
[dsp_cdiv](#), 241
[dsp_cdiv2](#), 241
[dsp_cdiv2f](#), 241
[dsp_cdiv3](#), 241
[dsp_cdiv3f](#), 241
[dsp_cdivf](#), 241
[dsp_cfft_nip](#), 260
[dsp_cfft_nip](#), 260
[dsp_chmul](#), 240
[dsp_chmul2](#), 240
[dsp_chmul2f](#), 240
[dsp_chmulf](#), 240
[dsp_cifft_nip](#), 260
[dsp_cifftf_nip](#), 260
[dsp_clip](#), 243
[dsp_clip2](#), 243
[dsp_clip2_nip](#), 243
[dsp_clip2f](#), 243
[dsp_clip2f_nip](#), 243
[dsp_clip_nip](#), 243
[dsp_clipf](#), 243
[dsp_clipf_nip](#), 243
[dsp_clipzero](#), 243
[dsp_clipzero_nip](#), 244
[dsp_clipzerof](#), 243
[dsp_clipzerof_nip](#), 244
[dsp_cmul](#), 240
[dsp_cmul2](#), 240
[dsp_cmul2f](#), 240
[dsp_cmul3](#), 240
[dsp_cmul3f](#), 240
[dsp_cmulf](#), 240
[dsp_convert64c](#), 249
[dsp_convert64f](#), 249
[dsp_convert64i](#), 249
[dsp_convert64s](#), 249
[dsp_convertf32](#), 249
[dsp_convertf32c](#), 249
[dsp_convertf32i](#), 249
[dsp_convertf32s](#), 249
[dsp_converts16](#), 249
[dsp_converts16f](#), 249
[dsp_converts32](#), 249
[dsp_converts32f](#), 249
[dsp_convertu8](#), 248
[dsp_convertu8f](#), 248
[dsp_convolve](#), 244
[dsp_convolve2](#), 244
[dsp_convolve2f](#), 244
[dsp_convolvef](#), 244
[dsp_copy](#), 244
[dsp_copyf](#), 244
[dsp_correlate](#), 244
[dsp_correlate2](#), 245
[dsp_correlate2f](#), 245
[dsp_correlatef](#), 244
[dsp_crosscorr](#), 250
[dsp_crosscorr2](#), 251
[dsp_crosscorr2f](#), 251
[dsp_crosscorr3](#), 251
[dsp_crosscorr3f](#), 251
[dsp_crosscorrf](#), 250
[dsp_cset](#), 243
[dsp_cset2](#), 243
[dsp_cset2f](#), 243
[dsp_csetf](#), 243
[dsp_csub](#), 238
[dsp_csub2](#), 238
[dsp_csub2f](#), 238
[dsp_csub3](#), 238
[dsp_csub3f](#), 238
[dsp_csubf](#), 238
[dsp_czero](#), 242
[dsp_czerof](#), 242
[dsp_dec_fft_delete](#), 261
[dsp_dec_fft_get](#), 262
[dsp_dec_fft_getf](#), 262
[dsp_dec_fft_init](#), 261
[dsp_dec_fft_initf](#), 261
[dsp_dec_fft_new](#), 261
[dsp_dec_fft_put](#), 262
[dsp_dec_fft_putf](#), 261
[dsp_dec_fft_uninit](#), 261
[dsp_dec_fir_delete](#), 262
[dsp_dec_fir_get](#), 263

- [dsp_dec_fir_getf, 263](#)
- [dsp_dec_fir_init, 262](#)
- [dsp_dec_fir_initf, 262](#)
- [dsp_dec_fir_new, 262](#)
- [dsp_dec_fir_put, 263](#)
- [dsp_dec_fir_putf, 262](#)
- [dsp_dec_fir_uninit, 262](#)
- [dsp_dec_iir_delete, 263](#)
- [dsp_dec_iir_get, 264](#)
- [dsp_dec_iir_getf, 264](#)
- [dsp_dec_iir_init, 263](#)
- [dsp_dec_iir_initf, 263](#)
- [dsp_dec_iir_new, 263](#)
- [dsp_dec_iir_put, 264](#)
- [dsp_dec_iir_putf, 263](#)
- [dsp_dec_iir_uninit, 263](#)
- [dsp_dec_rec_delete, 264](#)
- [dsp_dec_rec_get, 265](#)
- [dsp_dec_rec_getf, 265](#)
- [dsp_dec_rec_init, 264](#)
- [dsp_dec_rec_initf, 264](#)
- [dsp_dec_rec_new, 264](#)
- [dsp_dec_rec_put, 265](#)
- [dsp_dec_rec_putf, 264](#)
- [dsp_dec_rec_uninit, 264](#)
- [dsp_decimate, 252](#)
- [dsp_decimateavg, 252](#)
- [dsp_decimateavgf, 252](#)
- [dsp_decimatef, 252](#)
- [dsp_deg2rad, 258](#)
- [dsp_deg2radf, 258](#)
- [dsp_delete, 255](#)
- [dsp_div, 240](#)
- [dsp_div1x, 241](#)
- [dsp_div1x_nip, 241](#)
- [dsp_div1xf, 241](#)
- [dsp_div1xf_nip, 241](#)
- [dsp_div2, 241](#)
- [dsp_div2f, 241](#)
- [dsp_div3, 241](#)
- [dsp_div3f, 241](#)
- [dsp_divf, 240](#)
- [dsp_dotproduct, 245](#)
- [dsp_dotproductf, 245](#)
- [dsp_energy, 251](#)
- [dsp_energyf, 251](#)
- [dsp_extract, 254](#)
- [dsp_extractf, 254](#)
- [dsp_fft, 259](#)
- [dsp_fft_init, 259](#)

- [dsp_fft_nip, 259](#)
- [dsp_fft_uninit, 259](#)
- [dsp_fftf, 259](#)
- [dsp_fftf_nip, 259](#)
- [dsp_fftw_converted2d, 255](#)
- [dsp_fftw_converted2f, 255](#)
- [dsp_fftw_convertcf2d, 255](#)
- [dsp_fftw_convertcf2f, 255](#)
- [dsp_fftw_convertd2cd, 255](#)
- [dsp_fftw_convertd2cf, 255](#)
- [dsp_fftw_convertf2cd, 254](#)
- [dsp_fftw_convertf2cf, 254](#)
- [dsp_filter_delete, 269](#)
- [dsp_filter_design_bp, 272](#)
- [dsp_filter_design_bp2, 272](#)
- [dsp_filter_design_bp2f, 272](#)
- [dsp_filter_design_bpf, 272](#)
- [dsp_filter_design_br, 272](#)
- [dsp_filter_design_br2, 272](#)
- [dsp_filter_design_br2f, 272](#)
- [dsp_filter_design_brf, 272](#)
- [dsp_filter_design_hp, 271](#)
- [dsp_filter_design_hp2, 272](#)
- [dsp_filter_design_hp2f, 272](#)
- [dsp_filter_design_hpf, 271](#)
- [dsp_filter_design_lp, 271](#)
- [dsp_filter_design_lp2, 271](#)
- [dsp_filter_design_lp2f, 271](#)
- [dsp_filter_design_lpf, 271](#)
- [dsp_filter_get, 271](#)
- [dsp_filter_get_ccoeffs, 270](#)
- [dsp_filter_get_ccoeffsf, 270](#)
- [dsp_filter_get_coeffs, 270](#)
- [dsp_filter_get_coefssf, 270](#)
- [dsp_filter_gettf, 271](#)
- [dsp_filter_init, 269](#)
- [dsp_filter_init2, 269](#)
- [dsp_filter_init2f, 269](#)
- [dsp_filter_init_hp, 270](#)
- [dsp_filter_init_hpf, 270](#)
- [dsp_filter_init_lp, 269](#)
- [dsp_filter_init_lpf, 269](#)
- [dsp_filter_initf, 269](#)
- [dsp_filter_new, 269](#)
- [dsp_filter_put, 270](#)
- [dsp_filter_put2, 271](#)
- [dsp_filter_put2f, 271](#)
- [dsp_filter_putf, 270](#)
- [dsp_filter_set_ccoeffs, 270](#)
- [dsp_filter_set_ccoeffsf, 270](#)

[dsp_filter_set_coeffs, 270](#)
[dsp_filter_set_coefssf, 270](#)
[dsp_fir_allocate, 258](#)
[dsp_fir_allocatef, 258](#)
[dsp_fir_filter, 258](#)
[dsp_fir_filter_fst, 258](#)
[dsp_fir_filter_nip, 258](#)
[dsp_fir_filterf, 258](#)
[dsp_fir_filterf_fst, 258](#)
[dsp_fir_filterf_nip, 258](#)
[dsp_fir_free, 258](#)
[dsp_ifft_nip, 260](#)
[dsp_ifftf_nip, 260](#)
[dsp_iir_cas_clear, 261](#)
[dsp_iir_cas_delete, 260](#)
[dsp_iir_cas_init, 260](#)
[dsp_iir_cas_initf, 260](#)
[dsp_iir_cas_new, 260](#)
[dsp_iir_cas_process, 261](#)
[dsp_iir_cas_process_nip, 261](#)
[dsp_iir_cas_processf, 261](#)
[dsp_iir_cas_processf_nip, 261](#)
[dsp_iir_cas_uninit, 261](#)
[dsp_iir_clear, 259](#)
[dsp_iir_filter, 259](#)
[dsp_iir_filter_nip, 259](#)
[dsp_iir_filterf, 259](#)
[dsp_iir_filterf_nip, 259](#)
[dsp_iir_init, 259](#)
[dsp_iir_initf, 259](#)
[dsp_init, 236](#)
[dsp_int_fft_delete, 265](#)
[dsp_int_fft_get, 266](#)
[dsp_int_fft_getf, 266](#)
[dsp_int_fft_init, 265](#)
[dsp_int_fft_initf, 265](#)
[dsp_int_fft_new, 265](#)
[dsp_int_fft_put, 266](#)
[dsp_int_fft_putf, 265](#)
[dsp_int_fft_uninit, 265](#)
[dsp_int_fir_delete, 266](#)
[dsp_int_fir_get, 267](#)
[dsp_int_fir_getf, 267](#)
[dsp_int_fir_init, 266](#)
[dsp_int_fir_initf, 266](#)
[dsp_int_fir_new, 266](#)
[dsp_int_fir_put, 267](#)
[dsp_int_fir_putf, 266](#)
[dsp_int_fir_uninit, 266](#)
[dsp_int_iir_delete, 267](#)
[dsp_int_iir_get, 268](#)
[dsp_int_iir_getf, 268](#)
[dsp_int_iir_init, 267](#)
[dsp_int_iir_initf, 267](#)
[dsp_int_iir_new, 267](#)
[dsp_int_iir_put, 268](#)
[dsp_int_iir_putf, 267](#)
[dsp_int_iir_uninit, 267](#)
[dsp_int_rec_delete, 268](#)
[dsp_int_rec_get, 269](#)
[dsp_int_rec_getf, 269](#)
[dsp_int_rec_init, 268](#)
[dsp_int_rec_initf, 268](#)
[dsp_int_rec_new, 268](#)
[dsp_int_rec_put, 269](#)
[dsp_int_rec_putf, 268](#)
[dsp_int_rec_uninit, 268](#)
[dsp_interpolate, 252](#)
[dsp_interpolateavg, 253](#)
[dsp_interpolateavgf, 253](#)
[dsp_interpolatef, 252](#)
[dsp_magnitude, 251](#)
[dsp_magnitudef, 251](#)
[dsp_mean, 245](#)
[dsp_meanf, 245](#)
[dsp_median, 246](#)
[dsp_medianf, 246](#)
[dsp_minmax, 245](#)
[dsp_minmaxf, 245](#)
[dsp_mix, 253](#)
[dsp_mix2, 254](#)
[dsp_mix2f, 254](#)
[dsp_mixf, 253](#)
[dsp_mixn, 254](#)
[dsp_mixnf, 254](#)
[dsp_mul, 238](#)
[dsp_mul2, 240](#)
[dsp_mul2f, 240](#)
[dsp_mul3, 240](#)
[dsp_mul3f, 240](#)
[dsp_mul_nip, 240](#)
[dsp_muladd, 241](#)
[dsp_muladd_nip, 242](#)
[dsp_muladdf, 241](#)
[dsp_muladdf_nip, 242](#)
[dsp_mulf, 238](#)
[dsp_mulf_nip, 240](#)
[dsp_negate, 246](#)
[dsp_negate_nip, 246](#)
[dsp_negatef, 246](#)

- [dsp_negatef_nip, 246](#)
- [dsp_new, 255](#)
- [dsp_normalize, 246](#)
- [dsp_normalize_nip, 246](#)
- [dsp_normalizef, 246](#)
- [dsp_normalizef_nip, 246](#)
- [dsp_pack, 254](#)
- [dsp_packf, 254](#)
- [dsp_peaklevel, 253](#)
- [dsp_peaklevelf, 253](#)
- [dsp_phase, 251](#)
- [dsp_phasef, 251](#)
- [dsp_polar2cart, 250](#)
- [dsp_polar2cart2, 250](#)
- [dsp_polar2cart2f, 250](#)
- [dsp_polar2cart3, 250](#)
- [dsp_polar2cart3f, 250](#)
- [dsp_polar2cart4, 250](#)
- [dsp_polar2cart4f, 250](#)
- [dsp_polar2cartf, 250](#)
- [dsp_power, 251](#)
- [dsp_powerf, 251](#)
- [dsp_powerphase, 252](#)
- [dsp_powerphasef, 252](#)
- [dsp_product, 246](#)
- [dsp_productf, 246](#)
- [dsp_rad2deg, 258](#)
- [dsp_rad2degf, 258](#)
- [dsp_rebuf_clear, 273](#)
- [dsp_rebuf_copy, 273](#)
- [dsp_rebuf_delete, 272](#)
- [dsp_rebuf_get, 273](#)
- [dsp_rebuf_getf, 273](#)
- [dsp_rebuf_new, 272](#)
- [dsp_rebuf_put, 273](#)
- [dsp_rebuf_putf, 273](#)
- [dsp_rebuf_size, 273](#)
- [dsp_rebuffer, 257](#)
- [dsp_rebufferf, 257](#)
- [dsp_resample, 252](#)
- [dsp_resampleavg, 252](#)
- [dsp_resampleavgf, 252](#)
- [dsp_resamplef, 252](#)
- [dsp_reverse, 246](#)
- [dsp_reverse_nip, 247](#)
- [dsp_reversef, 246](#)
- [dsp_reversef_nip, 247](#)
- [dsp_rms, 253](#)
- [dsp_rmsf, 253](#)
- [dsp_round, 236](#)
- [dsp_roundf, 236](#)
- [dsp_scale, 247](#)
- [dsp_scale01, 247](#)
- [dsp_scale01_nip, 247](#)
- [dsp_scale01f, 247](#)
- [dsp_scale01f_nip, 247](#)
- [dsp_scale_nip, 247](#)
- [dsp_scalef, 247](#)
- [dsp_scalef_nip, 247](#)
- [dsp_set, 242](#)
- [dsp_set2, 243](#)
- [dsp_set2f, 243](#)
- [dsp_setf, 242](#)
- [dsp_sort, 247](#)
- [dsp_sortf, 247](#)
- [dsp_sortl, 247](#)
- [dsp_sqrt, 242](#)
- [dsp_sqrt_nip, 242](#)
- [dsp_sqrtf, 242](#)
- [dsp_sqrtf_nip, 242](#)
- [dsp_square, 248](#)
- [dsp_square_nip, 248](#)
- [dsp_squaref, 248](#)
- [dsp_squaref_nip, 248](#)
- [dsp_stddev, 247](#)
- [dsp_stddevf, 247](#)
- [dsp_sub, 237](#)
- [dsp_sub2, 238](#)
- [dsp_sub2f, 238](#)
- [dsp_sub3, 238](#)
- [dsp_sub3f, 238](#)
- [dsp_subf, 237](#)
- [dsp_sum, 248](#)
- [dsp_sumf, 248](#)
- [dsp_variance, 253](#)
- [dsp_variancef, 253](#)
- [dsp_win_bartlett, 255](#)
- [dsp_win_bartlettf, 255](#)
- [dsp_win_blackman, 255](#)
- [dsp_win_blackman_harris, 256](#)
- [dsp_win_blackman_harrisf, 256](#)
- [dsp_win_blackmanf, 255](#)
- [dsp_win_cos_tapered, 256](#)
- [dsp_win_cos_taperedf, 256](#)
- [dsp_win_dolph_chebyshev, 257](#)
- [dsp_win_dolph_chebyshevf, 257](#)
- [dsp_win_exact_blackman, 256](#)
- [dsp_win_exact_blackmanf, 256](#)
- [dsp_win_flat_top, 256](#)
- [dsp_win_flat_topf, 256](#)

- dsp_win_generic_cos, 256
- dsp_win_generic_cosf, 256
- dsp_win_hamming, 256
- dsp_win_hammingf, 256
- dsp_win_hanning, 257
- dsp_win_hanningf, 257
- dsp_win_kaiser, 257
- dsp_win_kaiser_bessel, 257
- dsp_win_kaiser_besself, 257
- dsp_win_kaiserf, 257
- dsp_win_tukey, 257
- dsp_win_tukeyf, 257
- dsp_zero, 242
- dsp_zeroof, 242
- cft1st
 - clTransform4, 199
 - clTransform8, 208
- cftb040
 - clTransformS, 212
- cftb1st
 - clTransformS, 212
- cftbsub
 - clTransform4, 199
 - clTransform8, 208
 - clTransformS, 212
- cftf040
 - clTransformS, 212
- cftf081
 - clTransformS, 212
- cftf082
 - clTransformS, 212
- cftf161
 - clTransformS, 212
- cftf162
 - clTransformS, 212
- cftf1st
 - clTransformS, 212
- cftfsub
 - clTransform4, 199
 - clTransform8, 208
 - clTransformS, 212
- cftfx41
 - clTransformS, 212
- cftleaf
 - clTransformS, 212
- cftmdl
 - clTransform4, 199
 - clTransform8, 208
- cftmdl1
 - clTransformS, 212
- cftmdl2
 - clTransformS, 212
- cftrec4
 - clTransformS, 212
- cfttree
 - clTransformS, 212
- cftx020
 - clTransformS, 212
- ChebyshevPolynom
 - clDSPOp, 50
- CheckSize
 - clReBuffer, 173
 - clReBufferT, 177
- clAlloc, 15
 - clAlloc, 16
- clAlloc
 - ~clAlloc, 17
 - bLocked, 21
 - clAlloc, 16
 - Copy, 18
 - CopyTo, 18
 - Free, 17
 - GetCopy, 19
 - GetPtr, 18
 - GetSize, 18
 - Lock, 18
 - lSize, 21
 - operator char *, 19
 - operator double *, 20
 - operator float *, 20
 - operator int *, 19
 - operator long *, 19
 - operator long double *, 20
 - operator short *, 19
 - operator unsigned char *, 19
 - operator unsigned int *, 19
 - operator unsigned long *, 19
 - operator unsigned short *, 19
 - operator void *, 20
 - operator=, 20
 - Resize, 17
 - Size, 17
 - UnLock, 18
 - vpPtr, 21
- clCondition, 22
 - clCondition, 22
- clCondition
 - ~clCondition, 22
 - clCondition, 22
 - Notify, 23

- NotifyAll, 23
- pthcCond, 23
- Wait, 23
- cIDSPAlloc, 24
 - cIDSPAlloc, 24
- cIDSPAlloc
 - cIDSPAlloc, 24
 - operator stDCplx *, 24
 - operator stDPolar *, 24
 - operator stSCplx *, 24
 - operator stSPolar *, 24
- cIDSPOp, 25
 - cIDSPOp, 40
- cIDSPOp
 - ~cIDSPOp, 40
 - Abs, 62
 - Add, 50–52
 - AutoCorrelate, 71
 - bFFTInitialized, 110
 - bRealTransform, 110
 - Cart2Polar, 40, 41
 - CartToPolar, 83, 84
 - ChebyshevPolynom, 50
 - cIDSPOp, 40
 - Clip, 65, 66
 - ClipZero, 67
 - Convert, 80–82
 - Convolve, 68, 69
 - Copy, 67, 68
 - Correlate, 69, 70
 - CplxAdd, 43
 - CplxConj, 49
 - CplxDiv, 46, 47
 - CplxExp, 47, 48
 - CplxLog, 48
 - CplxLog10, 48
 - CplxMul, 44, 45
 - CplxMulC, 46
 - CplxPow, 48
 - CplxRoot, 49
 - CplxSub, 44
 - CrossCorr, 85, 86
 - DBitRevWork, 110
 - DCosSinTable, 110
 - Decimate, 89
 - DecimateAvg, 89, 90
 - DegToRad, 102
 - DelCrossCorr, 86, 87
 - dFFFTScale, 110
 - Div, 58–60
 - Div1x, 60, 61
 - DotProduct, 72
 - dpCosSinTable, 110
 - dPI, 110
 - dpIIR_C, 110
 - dpIIR_X, 110
 - dpIIR_Y, 110
 - Energy, 87
 - Extract, 100, 101
 - fFFTSscale, 110
 - FFTBuF, 110
 - FFTi, 107
 - FFTInitialize, 106
 - FFTo, 107, 108
 - FFTUninitialize, 107
 - FFTWConvert, 103, 104
 - FIRAllocate, 104
 - FIRBuF, 110
 - FIRCoeff, 110
 - FIRFilter, 104, 105
 - FIRFilterF, 105
 - FIRFree, 105
 - FIRWork, 110
 - fpCosSinTable, 110
 - fPI, 110
 - fpIIR_C, 110
 - fpIIR_X, 110
 - fpIIR_Y, 110
 - IFFTo, 108
 - iFIRDlyIdx, 110
 - IIRClear, 106
 - IIRFilter, 106
 - IIRInitialize, 105
 - Interpolate, 90
 - InterpolateAvg, 90, 91
 - IFFTLength, 110
 - IFIRLength, 110
 - lpDBitRevWork, 110
 - lPrevDestCount, 110
 - lPrevSrcCount, 110
 - lpSBitRevWork, 110
 - Magnitude, 87, 88
 - Mean, 72, 73
 - Median, 73
 - MinMax, 72
 - Mix, 99, 100
 - ModZeroBessel, 49
 - Mul, 54–56
 - Mul2, 57, 58
 - MulAdd, 61, 62

- MulC, [57](#)
- Multiple, [49](#)
- Negate, [73](#), [74](#)
- Normalize, [74](#), [75](#)
- Pack, [101](#)
- PeakLevel, [92](#), [93](#)
- Phase, [88](#)
- Polar2Cart, [42](#), [43](#)
- PolarToCart, [84](#), [85](#)
- Power, [88](#)
- PowerPhase, [89](#)
- Product, [75](#)
- RadToDeg, [102](#)
- ReBuffer, [101](#), [102](#)
- Resample, [91](#)
- ResampleAvg, [91](#)
- Reverse, [75](#), [76](#)
- RMS, [91](#), [92](#)
- Round, [50](#)
- SBitRevWork, [110](#)
- Scale, [76](#), [77](#)
- Scale01, [77](#), [78](#)
- SCosSinTable, [110](#)
- Set, [64](#), [65](#)
- Sort, [78](#)
- Spatialize, [100](#)
- Sqrt, [62](#), [63](#)
- Square, [79](#), [80](#)
- StdDev, [78](#), [79](#)
- Sub, [52–54](#)
- Sum, [79](#)
- Tfrm, [110](#)
- Variance, [92](#)
- vpDTfrm, [110](#)
- vpSTfrm, [110](#)
- WinBartlett, [93](#)
- WinBlackman, [93](#), [94](#)
- WinBlackmanHarris, [94](#)
- WinCosTapered, [94](#)
- WinCosTaperedA, [95](#)
- WinDolphChebyshev, [98](#), [99](#)
- WinExactBlackman, [95](#)
- WinExp, [95](#), [96](#)
- WinFlatTop, [96](#)
- WinGenericCos, [96](#)
- WinHamming, [97](#)
- WinHanning, [97](#)
- WinKaiser, [97](#), [98](#)
- WinKaiserBessel, [98](#)
- WinTukey, [98](#)
- Zero, [63](#), [64](#)
- clDSPVector, [111](#)
 - clDSPVector, [115](#)
- clDSPVector
 - ~clDSPVector, [115](#)
 - Abs, [115](#)
 - AutoCorr, [115](#)
 - CartToPolar, [115](#)
 - clDSPVector, [115](#)
 - Clip, [115](#)
 - ClipZero, [115](#)
 - Conv, [115](#)
 - Convert, [115](#)
 - Corr, [115](#)
 - CrossCorr, [115](#)
 - Decimate, [115](#)
 - DecimateAvg, [115](#)
 - Div1x, [115](#)
 - DotProduct, [115](#)
 - DSP, [117](#)
 - Energy, [115](#)
 - Extract, [115](#)
 - FFT, [117](#)
 - FFTInitialize, [117](#)
 - FFTUninitialize, [117](#)
 - FFTWConvert, [115](#)
 - FIRAllocate, [115](#)
 - FIRFilter, [117](#)
 - FIRFree, [117](#)
 - IFFT, [117](#)
 - Interpolate, [115](#)
 - InterpolateAvg, [115](#)
 - IFFTSize, [117](#)
 - Magnitude, [115](#)
 - Mean, [115](#)
 - Median, [115](#)
 - MinMax, [115](#)
 - Mix, [115](#)
 - MulAdd, [115](#)
 - MulC, [115](#)
 - Negate, [115](#)
 - Normalize, [115](#)
 - operator *, [115](#)
 - operator *=[115](#)
 - operator+, [115](#)
 - operator+=[115](#)
 - operator-, [115](#)
 - operator-=[115](#)
 - operator/, [115](#)
 - operator/=[115](#)

- Pack, 115
- PeakLevel, 115
- Phase, 115
- PolarToCart, 115
- Power, 115
- PowerPhase, 115
- Ptr, 115
- Resample, 115
- ResampleAvg, 115
- Reverse, 115
- RMS, 115
- Scale, 115
- Scale01, 115
- Set, 115
- Sort, 115
- Sqrt, 115
- Square, 115
- StdDev, 115
- Sum, 115
- Variance, 115
- WinBartlett, 115
- WinBlackman, 115
- WinBlackmanHarris, 115
- WinCosTapered, 115
- WinDolphChebyshev, 115
- WinExactBlackman, 115
- WinExp, 115
- WinFlatTop, 115
- WinGenericCos, 115
- WinHamming, 115
- WinHanning, 115
- WinKaiser, 115
- WinKaiserBessel, 115
- WinTukey, 115
- Zero, 115
- clDynThreads, 118
 - clDynThreads, 119
- clDynThreads
 - clDynThreads, 119
 - Create, 119
 - InternalCaller, 119
 - Klass, 119
 - Method_t, 119
 - stParams2, 119
 - stpParams2, 119
- clDynThreads::_stParams2, 120
- clDynThreads::_stParams2
 - Method, 120
 - vpParam, 120
- clDynThreadsBase, 121
 - clDynThreadsBase, 122
- clDynThreadsBase
 - ~clDynThreadsBase, 122
 - clDynThreadsBase, 122
 - Create, 122
 - InternalCaller, 122
 - iThreadCount, 122
 - mapThreads, 122
 - MtxBase, 122
 - Self, 122
 - SetSched, 122
 - stParams, 122
 - stpParams, 122
 - Wait, 122
- clDynThreadsBase::_stParams, 123
- clDynThreadsBase::_stParams
 - Klass, 123
 - vpParam, 123
- Clear
 - clFlipBand, 153
 - clIIRCascade, 161
 - clReBuffer, 174
 - clReBufferT, 178
- clException, 124
 - clException, 125
- clException
 - ~clException, 125
 - clException, 125
 - iCode, 125
 - operator int, 125
 - operator std::string, 125
 - strMsg, 125
 - what, 125
- clFFFTDecimator, 126
 - clFFFTDecimator, 126
- clFFFTDecimator
 - ~clFFFTDecimator, 126
 - clFFFTDecimator, 126
 - DecBuf, 127
 - DSP, 127
 - Get, 127
 - Put, 126, 127
 - Uninitialize, 126
- clFFFTInterpolator, 128
 - clFFFTInterpolator, 128
- clFFFTInterpolator
 - ~clFFFTInterpolator, 128
 - clFFFTInterpolator, 128
 - DSP, 129
 - Get, 129

- IntBuf, 129
- Put, 128, 129
- Uninitialize, 128
- clFFTMultiRate, 130
 - clFFTMultiRate, 131
- clFFTMultiRate
 - ~clFFTMultiRate, 131
 - bInitialized, 131
 - clFFTMultiRate, 131
 - Filter, 131
 - Initialize, 131
 - IFactor, 131
 - IFilterSize, 131
 - Uninitialize, 131
- clFilter, 133
 - clFilter, 135
- clFilter
 - ~clFilter, 135
 - bInitialized, 143
 - CCoeffs, 143
 - clFilter, 135
 - CoeffWin, 143
 - CProc, 143
 - DesignBP, 141, 142
 - DesignBR, 142
 - DesignHP, 141
 - DesignLP, 140, 141
 - Get, 140
 - GetCoeffs, 139
 - GetDelay, 139
 - GetKaiserBeta, 136
 - InBuf, 143
 - InitCoeffsD, 135
 - InitCoeffsS, 135
 - InitDouble, 137
 - InitFloat, 137
 - Initialize, 136, 137
 - InitializeHP, 138
 - InitializeLP, 137
 - IFFTSize, 143
 - lHalfSize, 143
 - lNewSize, 143
 - lOldSize, 143
 - lSpectPoints, 143
 - OutBuf, 143
 - Prev, 143
 - Proc, 143
 - Put, 139, 140
 - ReadyFilterD, 136
 - ReadyFilterS, 136
 - SetCoeffs, 138
 - Uninitialize, 138
- clFIRDecimator, 144
 - clFIRDecimator, 144
- clFIRDecimator
 - ~clFIRDecimator, 144
 - clFIRDecimator, 144
 - DecBuf, 145
 - DSP, 145
 - Get, 145
 - InBuf, 145
 - Put, 145
 - Uninitialize, 144
- clFIRInterpolator, 146
 - clFIRInterpolator, 146
- clFIRInterpolator
 - ~clFIRInterpolator, 146
 - clFIRInterpolator, 146
 - DSP, 147
 - Get, 147
 - IntBuf, 147
 - OutBuf, 147
 - Put, 147
 - Uninitialize, 146
- clFIRMultiRate, 148
 - clFIRMultiRate, 149
- clFIRMultiRate
 - ~clFIRMultiRate, 149
 - clFIRMultiRate, 149
 - dGain, 149
 - fGain, 149
 - FIR, 149
 - Initialize, 149
 - IFactor, 149
 - Uninitialize, 149
- clFlipBand, 151
 - clFlipBand, 152
- clFlipBand
 - ~clFlipBand, 152
 - bInitialized, 153
 - Clear, 153
 - clFlipBand, 152
 - CProc, 153
 - DSP, 153
 - Get, 152, 153
 - InBuf, 153
 - Initialize, 152
 - lBlockSize, 153
 - lCBlockSize, 153
 - OutBuf, 153

- Proc, [153](#)
- Put, [152](#)
- Uninitialize, [152](#)
- clHankel, [154](#)
 - clHankel, [155](#)
- clHankel
 - ~clHankel, [155](#)
 - A, [158](#)
 - B0, [158](#)
 - B1, [158](#)
 - clHankel, [155](#)
 - DoAbel, [156](#)
 - dOutScale0, [158](#)
 - dOutScale1, [158](#)
 - DSP, [158](#)
 - fOutScale0, [158](#)
 - fOutScale1, [158](#)
 - GK, [158](#)
 - GX, [158](#)
 - InitAbel, [155](#)
 - Initialize, [156](#)
 - IFFTSize, [158](#)
 - lSize, [158](#)
 - Process0, [156](#), [157](#)
 - Process1, [157](#)
 - UninitAbel, [156](#)
 - Uninitialize, [156](#)
- clIIRCascade, [159](#)
 - clIIRCascade, [160](#)
- clIIRCascade
 - ~clIIRCascade, [160](#)
 - bInitialized, [161](#)
 - Clear, [161](#)
 - clIIRCascade, [160](#)
 - IIR, [161](#)
 - Initialize, [160](#)
 - lStages, [161](#)
 - Process, [160](#), [161](#)
 - Uninitialize, [160](#)
- clIIRDecimator, [162](#)
 - clIIRDecimator, [162](#)
- clIIRDecimator
 - ~clIIRDecimator, [162](#)
 - clIIRDecimator, [162](#)
 - DecBuf, [163](#)
 - DSP, [163](#)
 - Get, [163](#)
 - InBuf, [163](#)
 - Put, [163](#)
 - Uninitialize, [162](#)
- clIIRInterpolator, [164](#)
 - clIIRInterpolator, [164](#)
- clIIRInterpolator
 - ~clIIRInterpolator, [164](#)
 - clIIRInterpolator, [164](#)
 - DSP, [165](#)
 - Get, [165](#)
 - IntBuf, [165](#)
 - OutBuf, [165](#)
 - Put, [165](#)
 - Uninitialize, [164](#)
- clIIRMultiRate, [166](#)
 - clIIRMultiRate, [166](#)
- clIIRMultiRate
 - ~clIIRMultiRate, [166](#)
 - clIIRMultiRate, [166](#)
 - Initialize, [166](#), [167](#)
 - lFactor, [167](#)
 - Uninitialize, [167](#)
- Clip
 - clDSPOp, [65](#), [66](#)
 - clDSPVector, [115](#)
- ClipZero
 - clDSPOp, [67](#)
 - clDSPVector, [115](#)
- clMutex, [168](#)
 - clMutex, [168](#)
- clMutex
 - ~clMutex, [168](#)
 - clMutex, [168](#)
 - GetPtr, [169](#)
 - pthmMutex, [169](#)
 - Release, [169](#)
 - TryLock, [169](#)
 - Wait, [169](#)
- clPthCond, [170](#)
 - clPthCond, [170](#)
- clPthCond
 - ~clPthCond, [170](#)
 - clPthCond, [170](#)
 - Notify, [170](#)
 - pthcCond, [170](#)
 - Wait, [170](#)
- clPthMutex, [171](#)
 - clPthMutex, [171](#)
- clPthMutex
 - ~clPthMutex, [171](#)
 - clPthMutex, [171](#)
 - GetPtr, [171](#)
 - pthmMutex, [171](#)

- Release, 171
- Wait, 171
- clReBuffer, 172
 - clReBuffer, 173
- clReBuffer
 - ~clReBuffer, 173
 - Buffer, 174
 - CheckSize, 173
 - Clear, 174
 - clReBuffer, 173
 - Get, 173
 - GetCount, 174
 - GetPtr, 173
 - Index, 173
 - lCount, 174
 - lGetIndex, 174
 - lPutIndex, 174
 - lSize, 174
 - operator=, 174
 - Put, 173
- clReBufferT, 175
 - clReBufferT, 177
- clReBufferT
 - ~clReBufferT, 177
 - Buffer, 179
 - CheckSize, 177
 - Clear, 178
 - clReBufferT, 177
 - CopyGet, 177
 - Get, 177
 - GetCount, 178
 - GetPtr, 177
 - lCount, 179
 - lGetIndex, 179
 - lPutIndex, 179
 - lSize, 179
 - operator=, 178
 - operator[], 179
 - Put, 177
 - Resize, 178
 - SetSize, 178
 - Size, 178
- clRecDecimator, 180
 - clRecDecimator, 181
 - FILTER_TYPE_FFT, 181
 - FILTER_TYPE_FIR, 181
 - FILTER_TYPE_IIR, 181
- clRecDecimator
 - ~clRecDecimator, 181
 - bInitialized, 183
 - bpHalves, 183
 - clRecDecimator, 181
 - DecBuf, 184
 - eFilterType, 181
 - FFTDecBank, 184
 - FIRDecBank, 184
 - Get, 182, 183
 - IIRDecBank, 184
 - InitHalves, 181
 - Initialize, 181, 182
 - iType, 183
 - lDecSize, 183
 - lFactor, 183
 - lFilterSize, 183
 - lSubRounds, 183
 - Put, 182
 - Uninitialize, 182
- clRecInterpolator, 185
 - clRecInterpolator, 186
 - FILTER_TYPE_FFT, 186
 - FILTER_TYPE_FIR, 186
 - FILTER_TYPE_IIR, 186
- clRecInterpolator
 - ~clRecInterpolator, 186
 - bInitialized, 188
 - bpHalves, 189
 - clRecInterpolator, 186
 - eFilterType, 186
 - FFTIntBank, 189
 - FIRIntBank, 189
 - Get, 188
 - IIRIntBank, 189
 - InitHalves, 186
 - Initialize, 186, 187
 - IntBuf, 189
 - iType, 188
 - lFactor, 188
 - lFilterSize, 188
 - lIntSize, 188
 - lSubRounds, 188
 - Put, 187
 - Uninitialize, 187
- clRWLock, 190
 - clRWLock, 190
- clRWLock
 - ~clRWLock, 190
 - clRWLock, 190
 - ptrwlLock, 192
 - Release, 191
 - TryRead, 191

- TryWrite, 191
- WaitRead, 191
- WaitWrite, 191
- clSemaphore, 193
 - clSemaphore, 194
- clSemaphore
 - ~clSemaphore, 194
 - clSemaphore, 194
 - GetValue, 195
 - Initialize, 194
 - Post, 194
 - semSemaphore, 195
 - TryWait, 194
 - Wait, 194
- clTransform4, 196
- clTransform4
 - bitrv2, 199
 - bitrv2conj, 199
 - cdft, 199, 200
 - cft1st, 199
 - cftbsub, 199
 - cftfsub, 199
 - cftmdl, 199
 - dctsub, 199
 - ddct, 201, 202
 - ddst, 202, 203
 - dfct, 203
 - dfst, 203, 204
 - dstsub, 199
 - makect, 199
 - makewt, 199
 - rdft, 200, 201
 - rftbsub, 199
 - rftfsub, 199
- clTransform8, 205
- clTransform8
 - bitrv2, 208
 - bitrv2conj, 208
 - cdft, 208
 - cft1st, 208
 - cftbsub, 208
 - cftfsub, 208
 - cftmdl, 208
 - dctsub, 208
 - ddct, 208
 - ddst, 208
 - dfct, 208
 - dfst, 208
 - dstsub, 208
 - makect, 208
 - makewt, 208
 - rdft, 208
 - rftbsub, 208
 - rftfsub, 208
- clTransformS, 209
 - clTransformS, 212
- clTransformS
 - ~clTransformS, 212
 - bitrv2, 212
 - bitrv208, 212
 - bitrv208neg, 212
 - bitrv216, 212
 - bitrv216neg, 212
 - bitrv2conj, 212
 - cdft, 212
 - cftb040, 212
 - cftb1st, 212
 - cftbsub, 212
 - cftf040, 212
 - cftf081, 212
 - cftf082, 212
 - cftf161, 212
 - cftf162, 212
 - cftf1st, 212
 - cftfsub, 212
 - cftfx41, 212
 - cftleaf, 212
 - cftmdl1, 212
 - cftmdl2, 212
 - cftrec4, 212
 - cfttree, 212
 - cftx020, 212
 - clTransformS, 212
 - dctsub, 212
 - ddct, 212
 - ddst, 212
 - dfct, 212
 - dfst, 212
 - dstsub, 212
 - makect, 212
 - makeipt, 212
 - makewt, 212
 - rdft, 212
 - rftbsub, 212
 - rftfsub, 212
- clUserThreads, 213
 - clUserThreads, 213
- clUserThreads
 - bRun, 213
 - clUserThreads, 213

- Stop, 213
- Thread1, 213
- Thread2, 213
- Thread3, 213
- CoeffWin
 - clFilter, 143
- Compilers.hh, 274
 - likely, 274
 - prefetch, 274
 - unlikely, 274
- Condition.hh, 275
- CONSTFUNC
 - DSPOp.hh, 327
- Conv
 - clDSPVector, 115
- Convert
 - clDSPOp, 80–82
 - clDSPVector, 115
- Convolve
 - clDSPOp, 68, 69
- Copy
 - clAlloc, 18
 - clDSPOp, 67, 68
- CopyGet
 - clReBufferT, 177
- CopyTo
 - clAlloc, 18
- Corr
 - clDSPVector, 115
- Correlate
 - clDSPOp, 69, 70
- CplxAdd
 - clDSPOp, 43
- CplxConj
 - clDSPOp, 49
- CplxDiv
 - clDSPOp, 46, 47
- CplxExp
 - clDSPOp, 47, 48
- CplxLog
 - clDSPOp, 48
- CplxLog10
 - clDSPOp, 48
- CplxMul
 - clDSPOp, 44, 45
- CplxMulC
 - clDSPOp, 46
- CplxPow
 - clDSPOp, 48
- CplxRoot
 - clDSPOp, 49
- CplxSub
 - clDSPOp, 44
- CProc
 - clFilter, 143
 - clFlipBand, 153
- Create
 - clDynThreads, 119
 - clDynThreadsBase, 122
- CrossCorr
 - clDSPOp, 85, 86
 - clDSPVector, 115
- DBitRevWork
 - clDSPOp, 110
- DCosSinTable
 - clDSPOp, 110
- dctsub
 - clTransform4, 199
 - clTransform8, 208
 - clTransformS, 212
- ddct
 - clTransform4, 201, 202
 - clTransform8, 208
 - clTransformS, 212
- ddst
 - clTransform4, 202, 203
 - clTransform8, 208
 - clTransformS, 212
- DecBuf
 - clFFTDecimator, 127
 - clFIRDecimator, 145
 - clIIRDecimator, 163
 - clRecDecimator, 184
- Decimate
 - clDSPOp, 89
 - clDSPVector, 115
- DecimateAvg
 - clDSPOp, 89, 90
 - clDSPVector, 115
- DegToRad
 - clDSPOp, 102
- DelCrossCorr
 - clDSPOp, 86, 87
- DesignBP
 - clFilter, 141, 142
- DesignBR
 - clFilter, 142
- DesignHP
 - clFilter, 141

- DesignLP
 - clFilter, [140](#), [141](#)
- dfct
 - clTransform4, [203](#)
 - clTransform8, [208](#)
 - clTransformS, [212](#)
- dFFTScale
 - clDSPOp, [110](#)
- dfst
 - clTransform4, [203](#), [204](#)
 - clTransform8, [208](#)
 - clTransformS, [212](#)
- dGain
 - clFIRMultiRate, [149](#)
- Div
 - clDSPOp, [58–60](#)
- Div1x
 - clDSPOp, [60](#), [61](#)
 - clDSPVector, [115](#)
- DoAbel
 - clHankel, [156](#)
- DotProduct
 - clDSPOp, [72](#)
 - clDSPVector, [115](#)
- DoubleCompare
 - DSPOp.cc, [277](#)
- dOutScale0
 - clHankel, [158](#)
- dOutScale1
 - clHankel, [158](#)
- dpCosSinTable
 - clDSPOp, [110](#)
- dpH
 - Hankel.cc, [351](#)
- dPI
 - clDSPOp, [110](#)
- dpIIR_C
 - clDSPOp, [110](#)
- dpIIR_X
 - clDSPOp, [110](#)
- dpIIR_Y
 - clDSPOp, [110](#)
- dpLambda
 - Hankel.cc, [352](#)
- DSP
 - clDSPVector, [117](#)
 - clFFFTDecimator, [127](#)
 - clFFFTInterpolator, [129](#)
 - clFIRDecimator, [145](#)
 - clFIRInterpolator, [147](#)
 - clFlipBand, [153](#)
 - clHankel, [158](#)
 - clIIRDecimator, [163](#)
 - clIIRInterpolator, [165](#)
- dsp_abs
 - cdspop.cc, [242](#)
 - dspop.h, [300](#)
- dsp_abs_nip
 - cdspop.cc, [242](#)
 - dspop.h, [301](#)
- dsp_absf
 - cdspop.cc, [242](#)
 - dspop.h, [300](#)
- dsp_absf_nip
 - cdspop.cc, [242](#)
 - dspop.h, [301](#)
- dsp_add
 - cdspop.cc, [236](#)
 - dspop.h, [296](#)
- dsp_add2
 - cdspop.cc, [237](#)
 - dspop.h, [297](#)
- dsp_add2f
 - cdspop.cc, [237](#)
 - dspop.h, [297](#)
- dsp_add3
 - cdspop.cc, [237](#)
 - dspop.h, [297](#)
- dsp_add3f
 - cdspop.cc, [237](#)
 - dspop.h, [297](#)
- dsp_addf
 - cdspop.cc, [236](#)
 - dspop.h, [296](#)
- dsp_autocorr
 - cdspop.cc, [245](#)
 - dspop.h, [303](#)
- dsp_autocorr2
 - cdspop.cc, [245](#)
 - dspop.h, [303](#)
- dsp_autocorr2f
 - cdspop.cc, [245](#)
 - dspop.h, [303](#)
- dsp_autocorrf
 - cdspop.cc, [245](#)
 - dspop.h, [303](#)
- dsp_cadd
 - cdspop.cc, [237](#)
 - dspop.h, [297](#)
- dsp_cadd2

- cdspop.cc, [237](#)
- dspop.h, [297](#)
- dsp_cadd2f
 - cdspop.cc, [237](#)
 - dspop.h, [297](#)
- dsp_cadd3
 - cdspop.cc, [237](#)
 - dspop.h, [297](#)
- dsp_cadd3f
 - cdspop.cc, [237](#)
 - dspop.h, [297](#)
- dsp_caddf
 - cdspop.cc, [237](#)
 - dspop.h, [297](#)
- dsp_cart2polar
 - cdspop.cc, [249](#)
 - dspop.h, [306](#)
- dsp_cart2polar2
 - cdspop.cc, [250](#)
 - dspop.h, [306](#)
- dsp_cart2polar2f
 - cdspop.cc, [250](#)
 - dspop.h, [306](#)
- dsp_cart2polar3
 - cdspop.cc, [250](#)
 - dspop.h, [306](#)
- dsp_cart2polar3f
 - cdspop.cc, [250](#)
 - dspop.h, [306](#)
- dsp_cart2polar4
 - cdspop.cc, [250](#)
 - dspop.h, [306](#)
- dsp_cart2polar4f
 - cdspop.cc, [250](#)
 - dspop.h, [306](#)
- dsp_cart2polarf
 - cdspop.cc, [249](#)
 - dspop.h, [306](#)
- dsp_ccmul
 - cdspop.cc, [240](#)
 - dspop.h, [299](#)
- dsp_ccmul_nip
 - cdspop.cc, [240](#)
 - dspop.h, [299](#)
- dsp_ccmulf
 - cdspop.cc, [240](#)
 - dspop.h, [299](#)
- dsp_ccmulf_nip
 - cdspop.cc, [240](#)
 - dspop.h, [299](#)
- dsp_ccopy
 - cdspop.cc, [244](#)
- dsp_ccopyf
 - cdspop.cc, [244](#)
- dsp_cdiv
 - cdspop.cc, [241](#)
 - dspop.h, [300](#)
- dsp_cdiv2
 - cdspop.cc, [241](#)
 - dspop.h, [300](#)
- dsp_cdiv2f
 - cdspop.cc, [241](#)
 - dspop.h, [300](#)
- dsp_cdiv3
 - cdspop.cc, [241](#)
 - dspop.h, [300](#)
- dsp_cdiv3f
 - cdspop.cc, [241](#)
 - dspop.h, [300](#)
- dsp_cdivf
 - cdspop.cc, [241](#)
 - dspop.h, [300](#)
- dsp_cfft_nip
 - cdspop.cc, [260](#)
 - dspop.h, [315](#)
- dsp_cfftf_nip
 - cdspop.cc, [260](#)
 - dspop.h, [315](#)
- dsp_chmul
 - cdspop.cc, [240](#)
 - dspop.h, [299](#)
- dsp_chmul2
 - cdspop.cc, [240](#)
 - dspop.h, [299](#)
- dsp_chmul2f
 - cdspop.cc, [240](#)
 - dspop.h, [299](#)
- dsp_chmulf
 - cdspop.cc, [240](#)
 - dspop.h, [299](#)
- dsp_cifft_nip
 - cdspop.cc, [260](#)
 - dspop.h, [315](#)
- dsp_cifftf_nip
 - cdspop.cc, [260](#)
 - dspop.h, [315](#)
- dsp_clip
 - cdspop.cc, [243](#)
 - dspop.h, [301](#)
- dsp_clip2

- cdspop.cc, 243
- dspop.h, 302
- dsp_clip2_nip
 - cdspop.cc, 243
 - dspop.h, 302
- dsp_clip2f
 - cdspop.cc, 243
 - dspop.h, 302
- dsp_clip2f_nip
 - cdspop.cc, 243
 - dspop.h, 302
- dsp_clip_nip
 - cdspop.cc, 243
 - dspop.h, 302
- dsp_clipf
 - cdspop.cc, 243
 - dspop.h, 301
- dsp_clipf_nip
 - cdspop.cc, 243
 - dspop.h, 302
- dsp_clipzero
 - cdspop.cc, 243
 - dspop.h, 302
- dsp_clipzero_nip
 - cdspop.cc, 244
 - dspop.h, 302
- dsp_clipzerof
 - cdspop.cc, 243
 - dspop.h, 302
- dsp_clipzerof_nip
 - cdspop.cc, 244
 - dspop.h, 302
- dsp_cmul
 - cdspop.cc, 240
 - dspop.h, 299
- dsp_cmul2
 - cdspop.cc, 240
 - dspop.h, 299
- dsp_cmul2f
 - cdspop.cc, 240
 - dspop.h, 299
- dsp_cmul3
 - cdspop.cc, 240
 - dspop.h, 299
- dsp_cmul3f
 - cdspop.cc, 240
 - dspop.h, 299
- dsp_cmulf
 - cdspop.cc, 240
 - dspop.h, 299
- dsp_convert64c
 - cdspop.cc, 249
 - dspop.h, 306
- dsp_convert64f
 - cdspop.cc, 249
 - dspop.h, 306
- dsp_convert64i
 - cdspop.cc, 249
 - dspop.h, 306
- dsp_convert64s
 - cdspop.cc, 249
 - dspop.h, 306
- dsp_convertf32
 - cdspop.cc, 249
 - dspop.h, 306
- dsp_convertf32c
 - cdspop.cc, 249
 - dspop.h, 306
- dsp_convertf32i
 - cdspop.cc, 249
 - dspop.h, 306
- dsp_convertf32s
 - cdspop.cc, 249
 - dspop.h, 306
- dsp_converts16
 - cdspop.cc, 249
 - dspop.h, 306
- dsp_converts16f
 - cdspop.cc, 249
 - dspop.h, 306
- dsp_converts32
 - cdspop.cc, 249
 - dspop.h, 306
- dsp_converts32f
 - cdspop.cc, 249
 - dspop.h, 306
- dsp_convertu8
 - cdspop.cc, 248
 - dspop.h, 305
- dsp_convertu8f
 - cdspop.cc, 248
 - dspop.h, 305
- dsp_convolve
 - cdspop.cc, 244
 - dspop.h, 302
- dsp_convolve2
 - cdspop.cc, 244
 - dspop.h, 302
- dsp_convolve2f
 - cdspop.cc, 244

- dspop.h, 302
- dsp_convolvef
 - cdspop.cc, 244
 - dspop.h, 302
- dsp_copy
 - cdspop.cc, 244
 - dspop.h, 302
- dsp_copyf
 - cdspop.cc, 244
 - dspop.h, 302
- dsp_correlate
 - cdspop.cc, 244
 - dspop.h, 302
- dsp_correlate2
 - cdspop.cc, 245
 - dspop.h, 303
- dsp_correlate2f
 - cdspop.cc, 245
 - dspop.h, 303
- dsp_correlatef
 - cdspop.cc, 244
 - dspop.h, 302
- dsp_crosscorr
 - cdspop.cc, 250
 - dspop.h, 307
- dsp_crosscorr2
 - cdspop.cc, 251
 - dspop.h, 307
- dsp_crosscorr2f
 - cdspop.cc, 251
 - dspop.h, 307
- dsp_crosscorr3
 - cdspop.cc, 251
 - dspop.h, 307
- dsp_crosscorr3f
 - cdspop.cc, 251
 - dspop.h, 307
- dsp_crosscorrf
 - cdspop.cc, 250
 - dspop.h, 307
- dsp_cset
 - cdspop.cc, 243
 - dspop.h, 301
- dsp_cset2
 - cdspop.cc, 243
 - dspop.h, 301
- dsp_cset2f
 - cdspop.cc, 243
 - dspop.h, 301
- dsp_csetf
 - cdspop.cc, 243
 - dspop.h, 301
- cdspop.cc, 243
- dspop.h, 301
- dsp_csub
 - cdspop.cc, 238
 - dspop.h, 298
- dsp_csub2
 - cdspop.cc, 238
 - dspop.h, 298
- dsp_csub2f
 - cdspop.cc, 238
 - dspop.h, 298
- dsp_csub3
 - cdspop.cc, 238
 - dspop.h, 298
- dsp_csub3f
 - cdspop.cc, 238
 - dspop.h, 298
- dsp_csubf
 - cdspop.cc, 238
 - dspop.h, 298
- dsp_czero
 - cdspop.cc, 242
 - dspop.h, 301
- dsp_czerof
 - cdspop.cc, 242
 - dspop.h, 301
- dsp_dec_fft_delete
 - cdspop.cc, 261
 - dspop.h, 316
- dsp_dec_fft_get
 - cdspop.cc, 262
 - dspop.h, 317
- dsp_dec_fft_getf
 - cdspop.cc, 262
 - dspop.h, 317
- dsp_dec_fft_init
 - cdspop.cc, 261
 - dspop.h, 316
- dsp_dec_fft_initf
 - cdspop.cc, 261
 - dspop.h, 316
- dsp_dec_fft_new
 - cdspop.cc, 261
 - dspop.h, 316
- dsp_dec_fft_put
 - cdspop.cc, 262
 - dspop.h, 316
- dsp_dec_fft_putf
 - cdspop.cc, 261
 - dspop.h, 316

dsp_dec_fft_uninit
 cdspop.cc, 261
 dspop.h, 316
dsp_dec_fir_delete
 cdspop.cc, 262
 dspop.h, 317
dsp_dec_fir_get
 cdspop.cc, 263
 dspop.h, 317
dsp_dec_fir_getf
 cdspop.cc, 263
 dspop.h, 317
dsp_dec_fir_init
 cdspop.cc, 262
 dspop.h, 317
dsp_dec_fir_initf
 cdspop.cc, 262
 dspop.h, 317
dsp_dec_fir_new
 cdspop.cc, 262
 dspop.h, 317
dsp_dec_fir_put
 cdspop.cc, 263
 dspop.h, 317
dsp_dec_fir_putf
 cdspop.cc, 262
 dspop.h, 317
dsp_dec_fir_uninit
 cdspop.cc, 262
 dspop.h, 317
dsp_dec_iir_delete
 cdspop.cc, 263
 dspop.h, 318
dsp_dec_iir_get
 cdspop.cc, 264
 dspop.h, 318
dsp_dec_iir_getf
 cdspop.cc, 264
 dspop.h, 318
dsp_dec_iir_init
 cdspop.cc, 263
 dspop.h, 318
dsp_dec_iir_initf
 cdspop.cc, 263
 dspop.h, 318
dsp_dec_iir_new
 cdspop.cc, 263
 dspop.h, 318
dsp_dec_iir_put
 cdspop.cc, 264
 dspop.h, 318
dsp_dec_iir_putf
 cdspop.cc, 263
 dspop.h, 318
dsp_dec_iir_uninit
 cdspop.cc, 263
 dspop.h, 318
dsp_dec_rec_delete
 cdspop.cc, 264
 dspop.h, 318
dsp_dec_rec_get
 cdspop.cc, 265
 dspop.h, 319
dsp_dec_rec_getf
 cdspop.cc, 265
 dspop.h, 319
dsp_dec_rec_init
 cdspop.cc, 264
 dspop.h, 319
dsp_dec_rec_initf
 cdspop.cc, 264
 dspop.h, 319
dsp_dec_rec_new
 cdspop.cc, 264
 dspop.h, 318
dsp_dec_rec_put
 cdspop.cc, 265
 dspop.h, 319
dsp_dec_rec_putf
 cdspop.cc, 264
 dspop.h, 319
dsp_dec_rec_uninit
 cdspop.cc, 264
 dspop.h, 319
dsp_decfft_t
 dspop.h, 295
dsp_decfir_t
 dspop.h, 295
dsp_deciir_t
 dspop.h, 295
dsp_decimate
 cdspop.cc, 252
 dspop.h, 308
dsp_decimateavg
 cdspop.cc, 252
 dspop.h, 308
dsp_decimateavgf
 cdspop.cc, 252
 dspop.h, 308
dsp_decimatef

- cdspop.cc, 252
- dspop.h, 308
- dsp_decrec_t
 - dspop.h, 295
- dsp_deg2rad
 - cdspop.cc, 258
 - dspop.h, 313
- dsp_deg2radf
 - cdspop.cc, 258
 - dspop.h, 313
- dsp_delete
 - cdspop.cc, 255
 - dspop.h, 310
- dsp_div
 - cdspop.cc, 240
 - dspop.h, 299
- dsp_div1x
 - cdspop.cc, 241
 - dspop.h, 300
- dsp_div1x_nip
 - cdspop.cc, 241
 - dspop.h, 300
- dsp_div1xf
 - cdspop.cc, 241
 - dspop.h, 300
- dsp_div1xf_nip
 - cdspop.cc, 241
 - dspop.h, 300
- dsp_div2
 - cdspop.cc, 241
 - dspop.h, 300
- dsp_div2f
 - cdspop.cc, 241
 - dspop.h, 300
- dsp_div3
 - cdspop.cc, 241
 - dspop.h, 300
- dsp_div3f
 - cdspop.cc, 241
 - dspop.h, 300
- dsp_divf
 - cdspop.cc, 240
 - dspop.h, 299
- dsp_dotproduct
 - cdspop.cc, 245
 - dspop.h, 303
- dsp_dotproductf
 - cdspop.cc, 245
 - dspop.h, 303
- dsp_energy
 - cdspop.cc, 251
 - dspop.h, 307
- dsp_energyf
 - cdspop.cc, 251
 - dspop.h, 307
- dsp_extract
 - cdspop.cc, 254
 - dspop.h, 310
- dsp_extractf
 - cdspop.cc, 254
 - dspop.h, 310
- dsp_fft
 - cdspop.cc, 259
 - dspop.h, 314
- dsp_fft_init
 - cdspop.cc, 259
 - dspop.h, 314
- dsp_fft_nip
 - cdspop.cc, 259
 - dspop.h, 315
- dsp_fft_uninit
 - cdspop.cc, 259
 - dspop.h, 314
- dsp_fftf
 - cdspop.cc, 259
 - dspop.h, 314
- dsp_fftf_nip
 - cdspop.cc, 259
 - dspop.h, 315
- dsp_fftw_convertcd2d
 - cdspop.cc, 255
 - dspop.h, 310
- dsp_fftw_convertcd2f
 - cdspop.cc, 255
 - dspop.h, 310
- dsp_fftw_convertcf2d
 - cdspop.cc, 255
 - dspop.h, 310
- dsp_fftw_convertcf2f
 - cdspop.cc, 255
 - dspop.h, 310
- dsp_fftw_convertd2cd
 - cdspop.cc, 255
 - dspop.h, 310
- dsp_fftw_convertd2cf
 - cdspop.cc, 255
 - dspop.h, 310
- dsp_fftw_convertf2cd
 - cdspop.cc, 254
 - dspop.h, 310

- dsp_fftw_convertf2cf
 - cdspop.cc, [254](#)
 - dspop.h, [310](#)
- DSP_FILT_DEF_BETA
 - DSPConfig.hh, [276](#)
- DSP_FILT_DEF_BETAF
 - DSPConfig.hh, [276](#)
- DSP_FILT_DEF_OVERLAP
 - DSPConfig.hh, [276](#)
- DSP_FILT_DEF_OVERLAPF
 - DSPConfig.hh, [276](#)
- dsp_filter_delete
 - cdspop.cc, [269](#)
 - dspop.h, [323](#)
- dsp_filter_design_bp
 - cdspop.cc, [272](#)
 - dspop.h, [325](#)
- dsp_filter_design_bp2
 - cdspop.cc, [272](#)
 - dspop.h, [325](#)
- dsp_filter_design_bp2f
 - cdspop.cc, [272](#)
 - dspop.h, [325](#)
- dsp_filter_design_bpf
 - cdspop.cc, [272](#)
 - dspop.h, [325](#)
- dsp_filter_design_br
 - cdspop.cc, [272](#)
 - dspop.h, [325](#)
- dsp_filter_design_br2
 - cdspop.cc, [272](#)
 - dspop.h, [325](#)
- dsp_filter_design_br2f
 - cdspop.cc, [272](#)
 - dspop.h, [325](#)
- dsp_filter_design_brf
 - cdspop.cc, [272](#)
 - dspop.h, [325](#)
- dsp_filter_design_hp
 - cdspop.cc, [271](#)
 - dspop.h, [325](#)
- dsp_filter_design_hp2
 - cdspop.cc, [272](#)
 - dspop.h, [325](#)
- dsp_filter_design_hp2f
 - cdspop.cc, [272](#)
 - dspop.h, [325](#)
- dsp_filter_design_hpf
 - cdspop.cc, [271](#)
 - dspop.h, [325](#)
- dsp_filter_design_lp
 - cdspop.cc, [271](#)
 - dspop.h, [324](#)
- dsp_filter_design_lp2
 - cdspop.cc, [271](#)
 - dspop.h, [325](#)
- dsp_filter_design_lp2f
 - cdspop.cc, [271](#)
 - dspop.h, [325](#)
- dsp_filter_design_lpf
 - cdspop.cc, [271](#)
 - dspop.h, [324](#)
- dsp_filter_get
 - cdspop.cc, [271](#)
 - dspop.h, [324](#)
- dsp_filter_get_ccoeffs
 - cdspop.cc, [270](#)
 - dspop.h, [324](#)
- dsp_filter_get_ccoeffsf
 - cdspop.cc, [270](#)
 - dspop.h, [324](#)
- dsp_filter_get_coeffs
 - cdspop.cc, [270](#)
 - dspop.h, [324](#)
- dsp_filter_get_coeffsf
 - cdspop.cc, [270](#)
 - dspop.h, [324](#)
- dsp_filter_getf
 - cdspop.cc, [271](#)
 - dspop.h, [324](#)
- dsp_filter_init
 - cdspop.cc, [269](#)
 - dspop.h, [323](#)
- dsp_filter_init2
 - cdspop.cc, [269](#)
 - dspop.h, [323](#)
- dsp_filter_init2f
 - cdspop.cc, [269](#)
 - dspop.h, [323](#)
- dsp_filter_init_hp
 - cdspop.cc, [270](#)
 - dspop.h, [323](#)
- dsp_filter_init_hpf
 - cdspop.cc, [270](#)
 - dspop.h, [323](#)
- dsp_filter_init_lp
 - cdspop.cc, [269](#)
 - dspop.h, [323](#)
- dsp_filter_init_lpf
 - cdspop.cc, [269](#)

- dspop.h, 323
- dsp_filter_initf
 - cdspop.cc, 269
 - dspop.h, 323
- dsp_filter_new
 - cdspop.cc, 269
 - dspop.h, 323
- dsp_filter_put
 - cdspop.cc, 270
 - dspop.h, 324
- dsp_filter_put2
 - cdspop.cc, 271
 - dspop.h, 324
- dsp_filter_put2f
 - cdspop.cc, 271
 - dspop.h, 324
- dsp_filter_putf
 - cdspop.cc, 270
 - dspop.h, 324
- dsp_filter_set_ccoeffs
 - cdspop.cc, 270
 - dspop.h, 324
- dsp_filter_set_ccoeffsf
 - cdspop.cc, 270
 - dspop.h, 324
- dsp_filter_set_ceoffs
 - dspop.h, 324
- dsp_filter_set_coeffs
 - cdspop.cc, 270
- dsp_filter_set_coeffsf
 - cdspop.cc, 270
 - dspop.h, 324
- dsp_filter_t
 - dspop.h, 296
- dsp_fir_allocate
 - cdspop.cc, 258
 - dspop.h, 313
- dsp_fir_allocatef
 - cdspop.cc, 258
 - dspop.h, 313
- dsp_fir_filter
 - cdspop.cc, 258
 - dspop.h, 313
- dsp_fir_filterfst
 - cdspop.cc, 258
 - dspop.h, 314
- dsp_fir_filter_nip
 - cdspop.cc, 258
 - dspop.h, 314
- cdspop.cc, 258
- dspop.h, 313
- dsp_fir_filterfst
 - cdspop.cc, 258
 - dspop.h, 314
- dsp_fir_filter_nip
 - cdspop.cc, 258
 - dspop.h, 314
- dsp_fir_free
 - cdspop.cc, 258
 - dspop.h, 313
- dsp_iffit_nip
 - cdspop.cc, 260
 - dspop.h, 315
- dsp_iffitf_nip
 - cdspop.cc, 260
 - dspop.h, 315
- dsp_iir_cas_clear
 - cdspop.cc, 261
 - dspop.h, 316
- dsp_iir_cas_delete
 - cdspop.cc, 260
 - dspop.h, 315
- dsp_iir_cas_init
 - cdspop.cc, 260
 - dspop.h, 315
- dsp_iir_cas_initf
 - cdspop.cc, 260
 - dspop.h, 315
- dsp_iir_cas_new
 - cdspop.cc, 260
 - dspop.h, 315
- dsp_iir_cas_process
 - cdspop.cc, 261
 - dspop.h, 316
- dsp_iir_cas_process_nip
 - cdspop.cc, 261
 - dspop.h, 316
- dsp_iir_cas_processf
 - cdspop.cc, 261
 - dspop.h, 316
- dsp_iir_cas_processf_nip
 - cdspop.cc, 261
 - dspop.h, 316
- dsp_iir_cas_uninit
 - cdspop.cc, 261
 - dspop.h, 316
- dsp_iir_clear
 - cdspop.cc, 259
 - dspop.h, 314

- dsp_iir_filter
 - cdspop.cc, [259](#)
 - dspop.h, [314](#)
- dsp_iir_filter_nip
 - cdspop.cc, [259](#)
 - dspop.h, [314](#)
- dsp_iir_filterf
 - cdspop.cc, [259](#)
 - dspop.h, [314](#)
- dsp_iir_filterf_nip
 - cdspop.cc, [259](#)
 - dspop.h, [314](#)
- dsp_iir_init
 - cdspop.cc, [259](#)
 - dspop.h, [314](#)
- dsp_iir_initf
 - cdspop.cc, [259](#)
 - dspop.h, [314](#)
- dsp_iircas_t
 - dspop.h, [295](#)
- dsp_init
 - cdspop.cc, [236](#)
 - dspop.h, [296](#)
- dsp_int_fft_delete
 - cdspop.cc, [265](#)
 - dspop.h, [319](#)
- dsp_int_fft_get
 - cdspop.cc, [266](#)
 - dspop.h, [320](#)
- dsp_int_fft_getf
 - cdspop.cc, [266](#)
 - dspop.h, [320](#)
- dsp_int_fft_init
 - cdspop.cc, [265](#)
 - dspop.h, [320](#)
- dsp_int_fft_initf
 - cdspop.cc, [265](#)
 - dspop.h, [319](#)
- dsp_int_fft_new
 - cdspop.cc, [265](#)
 - dspop.h, [319](#)
- dsp_int_fft_put
 - cdspop.cc, [266](#)
 - dspop.h, [320](#)
- dsp_int_fft_putf
 - cdspop.cc, [265](#)
 - dspop.h, [320](#)
- dsp_int_fft_uninit
 - cdspop.cc, [265](#)
 - dspop.h, [320](#)
- dsp_int_fir_delete
 - cdspop.cc, [266](#)
 - dspop.h, [320](#)
- dsp_int_fir_get
 - cdspop.cc, [267](#)
 - dspop.h, [321](#)
- dsp_int_fir_getf
 - cdspop.cc, [267](#)
 - dspop.h, [321](#)
- dsp_int_fir_init
 - cdspop.cc, [266](#)
 - dspop.h, [320](#)
- dsp_int_fir_initf
 - cdspop.cc, [266](#)
 - dspop.h, [320](#)
- dsp_int_fir_new
 - cdspop.cc, [266](#)
 - dspop.h, [320](#)
- dsp_int_fir_put
 - cdspop.cc, [267](#)
 - dspop.h, [321](#)
- dsp_int_fir_putf
 - cdspop.cc, [266](#)
 - dspop.h, [321](#)
- dsp_int_fir_uninit
 - cdspop.cc, [266](#)
 - dspop.h, [321](#)
- dsp_int_iir_delete
 - cdspop.cc, [267](#)
 - dspop.h, [321](#)
- dsp_int_iir_get
 - cdspop.cc, [268](#)
 - dspop.h, [322](#)
- dsp_int_iir_getf
 - cdspop.cc, [268](#)
 - dspop.h, [322](#)
- dsp_int_iir_init
 - cdspop.cc, [267](#)
 - dspop.h, [321](#)
- dsp_int_iir_initf
 - cdspop.cc, [267](#)
 - dspop.h, [321](#)
- dsp_int_iir_new
 - cdspop.cc, [267](#)
 - dspop.h, [321](#)
- dsp_int_iir_put
 - cdspop.cc, [268](#)
 - dspop.h, [322](#)
- dsp_int_iir_putf
 - cdspop.cc, [267](#)

- dspop.h, 321
- dsp_int_iir_uninit
 - cdspop.cc, 267
 - dspop.h, 321
- dsp_int_rec_delete
 - cdspop.cc, 268
 - dspop.h, 322
- dsp_int_rec_get
 - cdspop.cc, 269
 - dspop.h, 323
- dsp_int_rec_getf
 - cdspop.cc, 269
 - dspop.h, 323
- dsp_int_rec_init
 - cdspop.cc, 268
 - dspop.h, 322
- dsp_int_rec_initf
 - cdspop.cc, 268
 - dspop.h, 322
- dsp_int_rec_new
 - cdspop.cc, 268
 - dspop.h, 322
- dsp_int_rec_put
 - cdspop.cc, 269
 - dspop.h, 322
- dsp_int_rec_putf
 - cdspop.cc, 268
 - dspop.h, 322
- dsp_int_rec_uninit
 - cdspop.cc, 268
 - dspop.h, 322
- dsp_interpolate
 - cdspop.cc, 252
 - dspop.h, 308
- dsp_interpolateavg
 - cdspop.cc, 253
 - dspop.h, 308
- dsp_interpolateavgf
 - cdspop.cc, 253
 - dspop.h, 308
- dsp_interpolatelf
 - cdspop.cc, 252
 - dspop.h, 308
- dsp_intfft_t
 - dspop.h, 295
- dsp_intfir_t
 - dspop.h, 295
- dsp_intiir_t
 - dspop.h, 296
- dsp_intrec_t
 - dspop.h, 296
- dsp_magnitude
 - cdspop.cc, 251
 - dspop.h, 307
- dsp_magnituf
 - cdspop.cc, 251
 - dspop.h, 307
- DSP_MAXBESSEL
 - DSPOp.hh, 327
- dsp_meadianf
 - dspop.h, 303
- dsp_mean
 - cdspop.cc, 245
 - dspop.h, 303
- dsp_meanf
 - cdspop.cc, 245
 - dspop.h, 303
- dsp_median
 - cdspop.cc, 246
 - dspop.h, 303
- dsp_medianf
 - cdspop.cc, 246
- dsp_minmax
 - cdspop.cc, 245
 - dspop.h, 303
- dsp_minmaxf
 - cdspop.cc, 245
 - dspop.h, 303
- dsp_mix
 - cdspop.cc, 253
 - dspop.h, 309
- dsp_mix2
 - cdspop.cc, 254
 - dspop.h, 310
- dsp_mix2f
 - cdspop.cc, 254
 - dspop.h, 310
- dsp_mixf
 - cdspop.cc, 253
 - dspop.h, 309
- dsp_mixn
 - cdspop.cc, 254
 - dspop.h, 310
- dsp_mixnf
 - cdspop.cc, 254
 - dspop.h, 310
- dsp_mul
 - cdspop.cc, 238
 - dspop.h, 298
- dsp_mul2

- cdspop.cc, 240
- dspop.h, 299
- dsp_mul2f
 - cdspop.cc, 240
 - dspop.h, 299
- dsp_mul3
 - cdspop.cc, 240
 - dspop.h, 299
- dsp_mul3f
 - cdspop.cc, 240
 - dspop.h, 299
- dsp_mul_nip
 - cdspop.cc, 240
 - dspop.h, 299
- dsp_muladd
 - cdspop.cc, 241
 - dspop.h, 300
- dsp_muladd_nip
 - cdspop.cc, 242
 - dspop.h, 300
- dsp_muladdf
 - cdspop.cc, 241
 - dspop.h, 300
- dsp_muladdf_nip
 - cdspop.cc, 242
 - dspop.h, 300
- dsp_mulf
 - cdspop.cc, 238
 - dspop.h, 298
- dsp_mulf_nip
 - cdspop.cc, 240
 - dspop.h, 299
- dsp_negate
 - cdspop.cc, 246
 - dspop.h, 303
- dsp_negate_nip
 - cdspop.cc, 246
 - dspop.h, 304
- dsp_negatef
 - cdspop.cc, 246
 - dspop.h, 303
- dsp_negatef_nip
 - cdspop.cc, 246
 - dspop.h, 304
- dsp_new
 - cdspop.cc, 255
 - dspop.h, 310
- dsp_normalize
 - cdspop.cc, 246
 - dspop.h, 304
- dsp_normalize_nip
 - cdspop.cc, 246
 - dspop.h, 304
- dsp_normalizef
 - cdspop.cc, 246
 - dspop.h, 304
- dsp_normalizef_nip
 - cdspop.cc, 246
 - dspop.h, 304
- dsp_pack
 - cdspop.cc, 254
 - dspop.h, 310
- dsp_packf
 - cdspop.cc, 254
 - dspop.h, 310
- dsp_peaklevel
 - cdspop.cc, 253
 - dspop.h, 309
- dsp_peaklevelf
 - cdspop.cc, 253
 - dspop.h, 309
- dsp_phase
 - cdspop.cc, 251
 - dspop.h, 308
- dsp_phasef
 - cdspop.cc, 251
 - dspop.h, 308
- dsp_polar2cart
 - cdspop.cc, 250
 - dspop.h, 306
- dsp_polar2cart2
 - cdspop.cc, 250
 - dspop.h, 307
- dsp_polar2cart2f
 - cdspop.cc, 250
 - dspop.h, 307
- dsp_polar2cart3
 - cdspop.cc, 250
 - dspop.h, 307
- dsp_polar2cart3f
 - cdspop.cc, 250
 - dspop.h, 307
- dsp_polar2cart4
 - cdspop.cc, 250
 - dspop.h, 307
- dsp_polar2cart4f
 - cdspop.cc, 250
 - dspop.h, 307
- dsp_polar2cartf
 - cdspop.cc, 250

- dspop.h, 306
- dsp_power
 - cdspop.cc, 251
 - dspop.h, 308
- dsp_powerf
 - cdspop.cc, 251
 - dspop.h, 308
- dsp_powerphase
 - cdspop.cc, 252
 - dspop.h, 308
- dsp_powerphasef
 - cdspop.cc, 252
 - dspop.h, 308
- dsp_product
 - cdspop.cc, 246
 - dspop.h, 304
- dsp_productf
 - cdspop.cc, 246
 - dspop.h, 304
- dsp_rad2deg
 - cdspop.cc, 258
 - dspop.h, 313
- dsp_rad2degf
 - cdspop.cc, 258
 - dspop.h, 313
- dsp_rebuf_clear
 - cdspop.cc, 273
 - dspop.h, 326
- dsp_rebuf_copy
 - cdspop.cc, 273
 - dspop.h, 326
- dsp_rebuf_delete
 - cdspop.cc, 272
 - dspop.h, 325
- dsp_rebuf_get
 - cdspop.cc, 273
 - dspop.h, 326
- dsp_rebuf_getf
 - cdspop.cc, 273
 - dspop.h, 326
- dsp_rebuf_new
 - cdspop.cc, 272
 - dspop.h, 325
- dsp_rebuf_put
 - cdspop.cc, 273
 - dspop.h, 326
- dsp_rebuf_putf
 - cdspop.cc, 273
 - dspop.h, 326
- dsp_rebuf_size
 - cdspop.cc, 273
 - dspop.h, 326
- dsp_rebuf_t
 - dspop.h, 296
- dsp_rebuffer
 - cdspop.cc, 257
 - dspop.h, 313
- dsp_rebufferf
 - cdspop.cc, 257
 - dspop.h, 313
- dsp_resample
 - cdspop.cc, 252
 - dspop.h, 309
- dsp_resampleavg
 - cdspop.cc, 252
 - dspop.h, 309
- dsp_resampleavgf
 - cdspop.cc, 252
 - dspop.h, 309
- dsp_resamplef
 - cdspop.cc, 252
 - dspop.h, 309
- dsp_reverse
 - cdspop.cc, 246
 - dspop.h, 304
- dsp_reverse_nip
 - cdspop.cc, 247
 - dspop.h, 304
- dsp_reversef
 - cdspop.cc, 246
 - dspop.h, 304
- dsp_reversef_nip
 - cdspop.cc, 247
 - dspop.h, 304
- dsp_rms
 - cdspop.cc, 253
 - dspop.h, 309
- dsp_rmsf
 - cdspop.cc, 253
 - dspop.h, 309
- dsp_round
 - cdspop.cc, 236
 - dspop.h, 296
- dsp_roundf
 - cdspop.cc, 236
 - dspop.h, 296
- dsp_scale
 - cdspop.cc, 247
 - dspop.h, 304
- dsp_scale01

- cdspop.cc, 247
- dspop.h, 304
- dsp_Scale01_nip
 - dspop.h, 305
- dsp_scale01_nip
 - cdspop.cc, 247
- dsp_scale01f
 - cdspop.cc, 247
 - dspop.h, 304
- dsp_scale01f_nip
 - cdspop.cc, 247
 - dspop.h, 305
- dsp_scale_nip
 - cdspop.cc, 247
 - dspop.h, 304
- dsp_scalef
 - cdspop.cc, 247
 - dspop.h, 304
- dsp_scalef_nip
 - cdspop.cc, 247
 - dspop.h, 304
- dsp_set
 - cdspop.cc, 242
 - dspop.h, 301
- dsp_set2
 - cdspop.cc, 243
 - dspop.h, 301
- dsp_set2f
 - cdspop.cc, 243
 - dspop.h, 301
- dsp_setf
 - cdspop.cc, 242
 - dspop.h, 301
- dsp_sort
 - cdspop.cc, 247
 - dspop.h, 305
- dsp_sortf
 - cdspop.cc, 247
 - dspop.h, 305
- dsp_sortl
 - cdspop.cc, 247
 - dspop.h, 305
- dsp_sqrt
 - cdspop.cc, 242
 - dspop.h, 301
- dsp_sqrt_nip
 - cdspop.cc, 242
 - dspop.h, 301
- dsp_sqrtf
 - cdspop.cc, 242
- dspop.h, 301
- dsp_sqrtf_nip
 - cdspop.cc, 242
 - dspop.h, 305
- dsp_square
 - cdspop.cc, 248
 - dspop.h, 305
- dsp_square_nip
 - cdspop.cc, 248
 - dspop.h, 305
- dsp_squaref
 - cdspop.cc, 248
 - dspop.h, 305
- dsp_squaref_nip
 - cdspop.cc, 248
 - dspop.h, 305
- dsp_stddev
 - cdspop.cc, 247
 - dspop.h, 305
- dsp_stddevf
 - cdspop.cc, 247
 - dspop.h, 305
- dsp_sub
 - cdspop.cc, 237
 - dspop.h, 297
- dsp_sub2
 - cdspop.cc, 238
 - dspop.h, 298
- dsp_sub2f
 - cdspop.cc, 238
 - dspop.h, 298
- dsp_sub3
 - cdspop.cc, 238
 - dspop.h, 298
- dsp_sub3f
 - cdspop.cc, 238
 - dspop.h, 298
- dsp_subf
 - cdspop.cc, 237
 - dspop.h, 297
- dsp_sum
 - cdspop.cc, 248
 - dspop.h, 305
- dsp_sumf
 - cdspop.cc, 248
 - dspop.h, 305
- dsp_t
 - dspop.h, 295
- dsp_variance
 - cdspop.cc, 253

- dspop.h, 309
- dsp_variancef
 - cdspop.cc, 253
 - dspop.h, 309
- dsp_win_bartlett
 - cdspop.cc, 255
 - dspop.h, 311
- dsp_win_bartlettf
 - cdspop.cc, 255
 - dspop.h, 311
- dsp_win_blackman
 - cdspop.cc, 255
 - dspop.h, 311
- dsp_win_blackman_harris
 - cdspop.cc, 256
 - dspop.h, 311
- dsp_win_blackman_harrisf
 - cdspop.cc, 256
 - dspop.h, 311
- dsp_win_blackmanf
 - cdspop.cc, 255
 - dspop.h, 311
- dsp_win_cos_tapered
 - cdspop.cc, 256
 - dspop.h, 311
- dsp_win_cos_taperedf
 - cdspop.cc, 256
 - dspop.h, 311
- dsp_win_dolph_chebyshev
 - cdspop.cc, 257
 - dspop.h, 313
- dsp_win_dolph_chebyshevf
 - cdspop.cc, 257
 - dspop.h, 313
- dsp_win_exact_blackman
 - cdspop.cc, 256
 - dspop.h, 311
- dsp_win_exact_blackmanf
 - cdspop.cc, 256
 - dspop.h, 311
- dsp_win_exp
 - dspop.h, 311
- dsp_win_expf
 - dspop.h, 311
- dsp_win_flat_top
 - cdspop.cc, 256
 - dspop.h, 312
- dsp_win_flat_topf
 - cdspop.cc, 256
 - dspop.h, 312
- dsp_win_generic_cos
 - cdspop.cc, 256
 - dspop.h, 312
- dsp_win_generic_cosf
 - cdspop.cc, 256
 - dspop.h, 312
- dsp_win_hamming
 - cdspop.cc, 256
 - dspop.h, 312
- dsp_win_hammingf
 - cdspop.cc, 256
 - dspop.h, 312
- dsp_win_hanning
 - cdspop.cc, 257
 - dspop.h, 312
- dsp_win_hanningf
 - cdspop.cc, 257
 - dspop.h, 312
- dsp_win_kaiser
 - cdspop.cc, 257
 - dspop.h, 312
- dsp_win_kaiser_bessel
 - cdspop.cc, 257
 - dspop.h, 312
- dsp_win_kaiser_besself
 - cdspop.cc, 257
 - dspop.h, 312
- dsp_win_kaiserf
 - cdspop.cc, 257
 - dspop.h, 312
- dsp_win_tukey
 - cdspop.cc, 257
 - dspop.h, 312
- dsp_win_tukeyf
 - cdspop.cc, 257
 - dspop.h, 312
- DSP_WISDOM_FILE
 - DSPOp.hh, 327
- dsp_zero
 - cdspop.cc, 242
 - dspop.h, 301
- dsp_zerof
 - cdspop.cc, 242
 - dspop.h, 301
- DSPConfig.hh, 276
 - DSP_FILT_DEF_BETA, 276
 - DSP_FILT_DEF_BETAF, 276
 - DSP_FILT_DEF_OVERLAP, 276

- DSP_FILT_DEF_OVERLAPF, 276
- DSPOp.cc, 277
 - DoubleCompare, 277
 - FloatCompare, 277
 - LongCompare, 277
- dspop.h, 278
 - dsp_abs, 300
 - dsp_abs_nip, 301
 - dsp_absf, 300
 - dsp_absf_nip, 301
 - dsp_add, 296
 - dsp_add2, 297
 - dsp_add2f, 297
 - dsp_add3, 297
 - dsp_add3f, 297
 - dsp_addf, 296
 - dsp_autocorr, 303
 - dsp_autocorr2, 303
 - dsp_autocorr2f, 303
 - dsp_autocorrf, 303
 - dsp_cadd, 297
 - dsp_cadd2, 297
 - dsp_cadd2f, 297
 - dsp_cadd3, 297
 - dsp_cadd3f, 297
 - dsp_caddf, 297
 - dsp_cart2polar, 306
 - dsp_cart2polar2, 306
 - dsp_cart2polar2f, 306
 - dsp_cart2polar3, 306
 - dsp_cart2polar3f, 306
 - dsp_cart2polar4, 306
 - dsp_cart2polar4f, 306
 - dsp_cart2polarf, 306
 - dsp_ccmul, 299
 - dsp_ccmul_nip, 299
 - dsp_ccmulf, 299
 - dsp_ccmulf_nip, 299
 - dsp_cdiv, 300
 - dsp_cdiv2, 300
 - dsp_cdiv2f, 300
 - dsp_cdiv3, 300
 - dsp_cdiv3f, 300
 - dsp_cdivf, 300
 - dsp_cfft_nip, 315
 - dsp_cfftf_nip, 315
 - dsp_chmul, 299
 - dsp_chmul2, 299
 - dsp_chmul2f, 299
 - dsp_chmulf, 299
 - dsp_cifft_nip, 315
 - dsp_cifftf_nip, 315
 - dsp_clip, 301
 - dsp_clip2, 302
 - dsp_clip2_nip, 302
 - dsp_clip2f, 302
 - dsp_clip2f_nip, 302
 - dsp_clip_nip, 302
 - dsp_clipf, 301
 - dsp_clipf_nip, 302
 - dsp_clipzero, 302
 - dsp_clipzero_nip, 302
 - dsp_clipzerof, 302
 - dsp_clipzerof_nip, 302
 - dsp_cmul, 299
 - dsp_cmul2, 299
 - dsp_cmul2f, 299
 - dsp_cmul3, 299
 - dsp_cmul3f, 299
 - dsp_cmulf, 299
 - dsp_convertdd64c, 306
 - dsp_convertdd64f, 306
 - dsp_convertdd64i, 306
 - dsp_convertdd64s, 306
 - dsp_convertf32, 306
 - dsp_convertf32c, 306
 - dsp_convertf32i, 306
 - dsp_convertf32s, 306
 - dsp_converts16, 306
 - dsp_converts16f, 306
 - dsp_converts32, 306
 - dsp_converts32f, 306
 - dsp_convertu8, 305
 - dsp_convertu8f, 305
 - dsp_convolve, 302
 - dsp_convolve2, 302
 - dsp_convolve2f, 302
 - dsp_convolvef, 302
 - dsp_copy, 302
 - dsp_copyf, 302
 - dsp_correlate, 302
 - dsp_correlate2, 303
 - dsp_correlate2f, 303
 - dsp_correlatef, 302
 - dsp_crosscorr, 307
 - dsp_crosscorr2, 307
 - dsp_crosscorr2f, 307
 - dsp_crosscorr3, 307
 - dsp_crosscorr3f, 307

dsp_crosscorr, 307
dsp_cset, 301
dsp_cset2, 301
dsp_cset2f, 301
dsp_csetf, 301
dsp_csub, 298
dsp_csub2, 298
dsp_csub2f, 298
dsp_csub3, 298
dsp_csub3f, 298
dsp_csubf, 298
dsp_czero, 301
dsp_czerof, 301
dsp_dec_fft_delete, 316
dsp_dec_fft_get, 317
dsp_dec_fft_getf, 317
dsp_dec_fft_init, 316
dsp_dec_fft_initf, 316
dsp_dec_fft_new, 316
dsp_dec_fft_put, 316
dsp_dec_fft_putf, 316
dsp_dec_fft_uninit, 316
dsp_dec_fir_delete, 317
dsp_dec_fir_get, 317
dsp_dec_fir_getf, 317
dsp_dec_fir_init, 317
dsp_dec_fir_initf, 317
dsp_dec_fir_new, 317
dsp_dec_fir_put, 317
dsp_dec_fir_putf, 317
dsp_dec_fir_uninit, 317
dsp_dec_iir_delete, 318
dsp_dec_iir_get, 318
dsp_dec_iir_getf, 318
dsp_dec_iir_init, 318
dsp_dec_iir_initf, 318
dsp_dec_iir_new, 318
dsp_dec_iir_put, 318
dsp_dec_iir_putf, 318
dsp_dec_iir_uninit, 318
dsp_dec_rec_delete, 318
dsp_dec_rec_get, 319
dsp_dec_rec_getf, 319
dsp_dec_rec_init, 319
dsp_dec_rec_initf, 319
dsp_dec_rec_new, 318
dsp_dec_rec_put, 319
dsp_dec_rec_putf, 319
dsp_dec_rec_uninit, 319
dsp_decfft_t, 295
dsp_decfir_t, 295
dsp_deciir_t, 295
dsp_decimate, 308
dsp_decimateavg, 308
dsp_decimateavgf, 308
dsp_decimatef, 308
dsp_decrec_t, 295
dsp_deg2rad, 313
dsp_deg2radf, 313
dsp_delete, 310
dsp_div, 299
dsp_div1x, 300
dsp_div1x_nip, 300
dsp_div1xf, 300
dsp_div1xf_nip, 300
dsp_div2, 300
dsp_div2f, 300
dsp_div3, 300
dsp_div3f, 300
dsp_divf, 299
dsp_dotproduct, 303
dsp_dotproductf, 303
dsp_energy, 307
dsp_energyf, 307
dsp_extract, 310
dsp_extractf, 310
dsp_fft, 314
dsp_fft_init, 314
dsp_fft_nip, 315
dsp_fft_uninit, 314
dsp_fftf, 314
dsp_fftf_nip, 315
dsp_fftw_converted2d, 310
dsp_fftw_converted2f, 310
dsp_fftw_convertcf2d, 310
dsp_fftw_convertcf2f, 310
dsp_fftw_convertcd2d, 310
dsp_fftw_convertcd2f, 310
dsp_fftw_convertf2cd, 310
dsp_fftw_convertf2cf, 310
dsp_filter_delete, 323
dsp_filter_design_bp, 325
dsp_filter_design_bp2, 325
dsp_filter_design_bp2f, 325
dsp_filter_design_bpf, 325
dsp_filter_design_br, 325
dsp_filter_design_br2, 325
dsp_filter_design_br2f, 325
dsp_filter_design_brf, 325
dsp_filter_design_hp, 325

- [dsp_filter_design_hp2, 325](#)
- [dsp_filter_design_hp2f, 325](#)
- [dsp_filter_design_hpf, 325](#)
- [dsp_filter_design_lp, 324](#)
- [dsp_filter_design_lp2, 325](#)
- [dsp_filter_design_lp2f, 325](#)
- [dsp_filter_design_lpf, 324](#)
- [dsp_filter_get, 324](#)
- [dsp_filter_get_ccoeffs, 324](#)
- [dsp_filter_get_ccoeffsf, 324](#)
- [dsp_filter_get_ccoeffs, 324](#)
- [dsp_filter_get_ccoeffsf, 324](#)
- [dsp_filter_getf, 324](#)
- [dsp_filter_init, 323](#)
- [dsp_filter_init2, 323](#)
- [dsp_filter_init2f, 323](#)
- [dsp_filter_init_hp, 323](#)
- [dsp_filter_init_hpf, 323](#)
- [dsp_filter_init_lp, 323](#)
- [dsp_filter_init_lpf, 323](#)
- [dsp_filter_initf, 323](#)
- [dsp_filter_new, 323](#)
- [dsp_filter_put, 324](#)
- [dsp_filter_put2, 324](#)
- [dsp_filter_put2f, 324](#)
- [dsp_filter_putf, 324](#)
- [dsp_filter_set_ccoeffs, 324](#)
- [dsp_filter_set_ccoeffsf, 324](#)
- [dsp_filter_set_ccoeffs, 324](#)
- [dsp_filter_set_ccoeffsf, 324](#)
- [dsp_filter_t, 296](#)
- [dsp_fir_allocate, 313](#)
- [dsp_fir_allocatef, 313](#)
- [dsp_fir_filter, 313](#)
- [dsp_fir_filter_fst, 314](#)
- [dsp_fir_filter_nip, 314](#)
- [dsp_fir_filterf, 313](#)
- [dsp_fir_filterf_fst, 314](#)
- [dsp_fir_filterf_nip, 314](#)
- [dsp_fir_free, 313](#)
- [dsp_iff_t_nip, 315](#)
- [dsp_iff_t_nip, 315](#)
- [dsp_iir_cas_clear, 316](#)
- [dsp_iir_cas_delete, 315](#)
- [dsp_iir_cas_init, 315](#)
- [dsp_iir_cas_initf, 315](#)
- [dsp_iir_cas_new, 315](#)
- [dsp_iir_cas_process, 316](#)
- [dsp_iir_cas_process_nip, 316](#)
- [dsp_iir_cas_processf, 316](#)
- [dsp_iir_cas_processf_nip, 316](#)
- [dsp_iir_cas_uninit, 316](#)
- [dsp_iir_clear, 314](#)
- [dsp_iir_filter, 314](#)
- [dsp_iir_filter_nip, 314](#)
- [dsp_iir_filterf, 314](#)
- [dsp_iir_filterf_nip, 314](#)
- [dsp_iir_init, 314](#)
- [dsp_iir_initf, 314](#)
- [dsp_iircas_t, 295](#)
- [dsp_init, 296](#)
- [dsp_int_fft_delete, 319](#)
- [dsp_int_fft_get, 320](#)
- [dsp_int_fft_getf, 320](#)
- [dsp_int_fft_init, 320](#)
- [dsp_int_fft_initf, 319](#)
- [dsp_int_fft_new, 319](#)
- [dsp_int_fft_put, 320](#)
- [dsp_int_fft_putf, 320](#)
- [dsp_int_fft_uninit, 320](#)
- [dsp_int_fir_delete, 320](#)
- [dsp_int_fir_get, 321](#)
- [dsp_int_fir_getf, 321](#)
- [dsp_int_fir_init, 320](#)
- [dsp_int_fir_initf, 320](#)
- [dsp_int_fir_new, 320](#)
- [dsp_int_fir_put, 321](#)
- [dsp_int_fir_putf, 321](#)
- [dsp_int_fir_uninit, 321](#)
- [dsp_int_iir_delete, 321](#)
- [dsp_int_iir_get, 322](#)
- [dsp_int_iir_getf, 322](#)
- [dsp_int_iir_init, 321](#)
- [dsp_int_iir_initf, 321](#)
- [dsp_int_iir_new, 321](#)
- [dsp_int_iir_put, 322](#)
- [dsp_int_iir_putf, 321](#)
- [dsp_int_iir_uninit, 321](#)
- [dsp_int_rec_delete, 322](#)
- [dsp_int_rec_get, 323](#)
- [dsp_int_rec_getf, 323](#)
- [dsp_int_rec_init, 322](#)
- [dsp_int_rec_initf, 322](#)
- [dsp_int_rec_new, 322](#)
- [dsp_int_rec_put, 322](#)
- [dsp_int_rec_putf, 322](#)
- [dsp_int_rec_uninit, 322](#)
- [dsp_interpolate, 308](#)
- [dsp_interpolateavg, 308](#)
- [dsp_interpolateavgf, 308](#)

dsp_interpolatef, 308
dsp_intfft_t, 295
dsp_intfir_t, 295
dsp_intiir_t, 296
dsp_intrec_t, 296
dsp_magnitude, 307
dsp_magnitudef, 307
dsp_meadianf, 303
dsp_mean, 303
dsp_meanf, 303
dsp_median, 303
dsp_minmax, 303
dsp_minmaxf, 303
dsp_mix, 309
dsp_mix2, 310
dsp_mix2f, 310
dsp_mixf, 309
dsp_mixn, 310
dsp_mixnf, 310
dsp_mul, 298
dsp_mul2, 299
dsp_mul2f, 299
dsp_mul3, 299
dsp_mul3f, 299
dsp_mul_nip, 299
dsp_muladd, 300
dsp_muladd_nip, 300
dsp_muladdf, 300
dsp_muladdf_nip, 300
dsp_mulf, 298
dsp_mulf_nip, 299
dsp_negate, 303
dsp_negate_nip, 304
dsp_negatef, 303
dsp_negatef_nip, 304
dsp_new, 310
dsp_normalize, 304
dsp_normalize_nip, 304
dsp_normalizef, 304
dsp_normalizef_nip, 304
dsp_pack, 310
dsp_packf, 310
dsp_peaklevel, 309
dsp_peaklevelf, 309
dsp_phase, 308
dsp_phasef, 308
dsp_polar2cart, 306
dsp_polar2cart2, 307
dsp_polar2cart2f, 307
dsp_polar2cart3, 307
dsp_polar2cart3f, 307
dsp_polar2cart4, 307
dsp_polar2cart4f, 307
dsp_polar2cartf, 306
dsp_power, 308
dsp_powerf, 308
dsp_powerphase, 308
dsp_powerphasef, 308
dsp_product, 304
dsp_productf, 304
dsp_rad2deg, 313
dsp_rad2degf, 313
dsp_rebuf_clear, 326
dsp_rebuf_copy, 326
dsp_rebuf_delete, 325
dsp_rebuf_get, 326
dsp_rebuf_getf, 326
dsp_rebuf_new, 325
dsp_rebuf_put, 326
dsp_rebuf_putf, 326
dsp_rebuf_size, 326
dsp_rebuf_t, 296
dsp_rebuffer, 313
dsp_rebufferf, 313
dsp_resample, 309
dsp_resampleavg, 309
dsp_resampleavgf, 309
dsp_resamplef, 309
dsp_reverse, 304
dsp_reverse_nip, 304
dsp_reversef, 304
dsp_reversef_nip, 304
dsp_rms, 309
dsp_rmsf, 309
dsp_round, 296
dsp_roundf, 296
dsp_scale, 304
dsp_scale01, 304
dsp_Scale01_nip, 305
dsp_scale01f, 304
dsp_scale01f_nip, 305
dsp_scale_nip, 304
dsp_scalef, 304
dsp_scalef_nip, 304
dsp_set, 301
dsp_set2, 301
dsp_set2f, 301
dsp_setf, 301
dsp_sort, 305
dsp_sortf, 305

- dsp_sortl, 305
- dsp_sqrt, 301
- dsp_sqrt_nip, 301
- dsp_sqrtf, 301
- dsp_sqrtf_nip, 301
- dsp_square, 305
- dsp_square_nip, 305
- dsp_squaref, 305
- dsp_squaref_nip, 305
- dsp_stddev, 305
- dsp_stddevf, 305
- dsp_sub, 297
- dsp_sub2, 298
- dsp_sub2f, 298
- dsp_sub3, 298
- dsp_sub3f, 298
- dsp_subf, 297
- dsp_sum, 305
- dsp_sumf, 305
- dsp_t, 295
- dsp_variance, 309
- dsp_variancef, 309
- dsp_win_bartlett, 311
- dsp_win_bartlettf, 311
- dsp_win_blackman, 311
- dsp_win_blackman_harris, 311
- dsp_win_blackman_harrisf, 311
- dsp_win_blackmanf, 311
- dsp_win_cos_tapered, 311
- dsp_win_cos_taperedf, 311
- dsp_win_dolph_chebyshev, 313
- dsp_win_dolph_chebyshevf, 313
- dsp_win_exact_blackman, 311
- dsp_win_exact_blackmanf, 311
- dsp_win_exp, 311
- dsp_win_expf, 311
- dsp_win_flat_top, 312
- dsp_win_flat_topf, 312
- dsp_win_generic_cos, 312
- dsp_win_generic_cosf, 312
- dsp_win_hamming, 312
- dsp_win_hammingf, 312
- dsp_win_hanning, 312
- dsp_win_hanningf, 312
- dsp_win_kaiser, 312
- dsp_win_kaiser_bessel, 312
- dsp_win_kaiser_besself, 312
- dsp_win_kaiserf, 312
- dsp_win_tukey, 312
- dsp_win_tukeyf, 312
- dsp_zero, 301
- dsp_zerof, 301
- DSPOp.hh, 327
 - CONSTFUNC, 327
 - DSP_MAXBESSEL, 327
 - DSP_WISDOM_FILE, 327
 - INLINE, 327
- dsptypes.h, 328
 - stDCplx, 329
 - stDPolar, 330
 - stpDCplx, 329
 - stpDPolar, 330
 - stpSCplx, 329
 - stpSPolar, 329
 - stSCplx, 329
 - stSPolar, 329
 - utDCoord, 330
 - utpDCoord, 330
 - utpSCoord, 330
 - utSCoord, 330
- DSPV_SHORTER
 - DSPVector.hh, 331
- DSPV_SHORTER2
 - DSPVector.hh, 331
- DSPVector.hh, 331
 - DSPV_SHORTER, 331
 - DSPV_SHORTER2, 331
- dstsub
 - clTransform4, 199
 - clTransform8, 208
 - clTransformS, 212
- DynThreads.cc, 332
- DynThreads.cc
 - WrapDynThreadBase, 332
- DynThreads.hh, 333
- DynThreads.hh
 - ThreadMap_t, 333
- eFilterType
 - clRecDecimator, 181
 - clRecInterpolator, 186
- Energy
 - clDSPOp, 87
 - clDSPVector, 115
- Exception.hh, 334
- Extract
 - clDSPOp, 100, 101
 - clDSPVector, 115
- ffFTScale

- cIDSPOp, 110
- FFT
 - cIDSPVector, 117
- FFTBuf
 - cIDSPOp, 110
- FFTDecBank
 - clRecDecimator, 184
- FFTDecimator.cc, 335
- FFTDecimator.hh, 336
- FFTi
 - cIDSPOp, 107
- FFTInitialize
 - cIDSPOp, 106
 - cIDSPVector, 117
- FFTIntBank
 - clRecInterpolator, 189
- FFTInterpolator.cc, 337
- FFTInterpolator.hh, 338
- FFTMultiRate.cc, 339
- FFTMultiRate.hh, 340
- FFTMultiRate.hh
 - FFTMULTIRATE_-
 DELTAOMEGA, 340
 - FFTMULTIRATE_OVERLAP,
 340
 - FFTMULTIRATE_-
 RIPPLERATIO, 340
- FFTMULTIRATE_DELTAOMEGA
 - FFTMultiRate.hh, 340
- FFTMULTIRATE_OVERLAP
 - FFTMultiRate.hh, 340
- FFTMULTIRATE_RIPPLERATIO
 - FFTMultiRate.hh, 340
- FFTTo
 - cIDSPOp, 107, 108
- FFTUninitialize
 - cIDSPOp, 107
 - cIDSPVector, 117
- FFTWConvert
 - cIDSPOp, 103, 104
 - cIDSPVector, 115
- fGain
 - clFIRMultiRate, 149
- Filter
 - clFFTMultiRate, 131
- Filter.cc, 341
- Filter.hh, 342
 - FILTER_SMOOTH_DOLPH_-
 CHEBYSHEV, 342
 - FILTER_SMOOTH_KAISER,
 342
 - FILTER_SMOOTH_KAISER_-
 BESSEL, 342
 - FILTER_SMOOTH_NONE, 342
 - FILTER_SMOOTH_DOLPH_-
 CHEBYSHEV
 Filter.hh, 342
 - FILTER_SMOOTH_KAISER
 Filter.hh, 342
 - FILTER_SMOOTH_KAISER_-
 BESSEL
 Filter.hh, 342
 - FILTER_SMOOTH_NONE
 Filter.hh, 342
- FILTER_TYPE_FFT
 - clRecDecimator, 181
 - clRecInterpolator, 186
- FILTER_TYPE_FIR
 - clRecDecimator, 181
 - clRecInterpolator, 186
- FILTER_TYPE_IIR
 - clRecDecimator, 181
 - clRecInterpolator, 186
- FIR
 - clFIRMultiRate, 149
- FIRAllocate
 - cIDSPOp, 104
 - cIDSPVector, 115
- FIRBuf
 - cIDSPOp, 110
- FIRCoeff
 - cIDSPOp, 110
- FIRDecBank
 - clRecDecimator, 184
- FIRDecimator.cc, 343
- FIRDecimator.hh, 344
- FIRFilter
 - cIDSPOp, 104, 105
 - cIDSPVector, 117
- FIRFilterF
 - cIDSPOp, 105
- FIRFree
 - cIDSPOp, 105
 - cIDSPVector, 117
- FIRIntBank
 - clRecInterpolator, 189
- FIRInterpolator.cc, 345
- FIRInterpolator.hh, 346
- FIRMultiRate.cc, 347

- FIRMultiRate.hh, 348
- FIRWork
 - clDSPOp, 110
- FlipBand.cc, 349
- FlipBand.hh, 350
- FloatCompare
 - DSPOp.cc, 277
- fOutScale0
 - clHankel, 158
- fOutScale1
 - clHankel, 158
- fpCosSinTable
 - clDSPOp, 110
- fpH
 - Hankel.cc, 351
- fPI
 - clDSPOp, 110
- fpIIR_C
 - clDSPOp, 110
- fpIIR_X
 - clDSPOp, 110
- fpIIR_Y
 - clDSPOp, 110
- fpLambda
 - Hankel.cc, 352
- Free
 - clAlloc, 17
- Get
 - clFFFTDecimator, 127
 - clFFFTInterpolator, 129
 - clFilter, 140
 - clFIRDecimator, 145
 - clFIRInterpolator, 147
 - clFlipBand, 152, 153
 - clIIRDecimator, 163
 - clIIRInterpolator, 165
 - clReBuffer, 173
 - clReBufferT, 177
 - clRecDecimator, 182, 183
 - clRecInterpolator, 188
- GetCoeffs
 - clFilter, 139
- GetCopy
 - clAlloc, 19
- GetCount
 - clReBuffer, 174
 - clReBufferT, 178
- GetDelay
 - clFilter, 139
- GetKaiserBeta
 - clFilter, 136
- GetPtr
 - clAlloc, 18
 - clMutex, 169
 - clPthMutex, 171
 - clReBuffer, 173
 - clReBufferT, 177
- GetSize
 - clAlloc, 18
- GetValue
 - clSemaphore, 195
- GK
 - clHankel, 158
- GX
 - clHankel, 158
- Hankel.cc, 351
 - ABEL_NSE, 351
 - dpH, 351
 - dpLambda, 352
 - fpH, 351
 - fpLambda, 352
- Hankel.hh, 353
- I
 - _sDCplx, 9
 - _sSCplx, 11
- iCode
 - clException, 125
- IFFT
 - clDSPVector, 117
- IFFTo
 - clDSPOp, 108
- iFIRDlyIdx
 - clDSPOp, 110
- IIR
 - clIIRCascade, 161
- IIRCascade.cc, 354
- IIRCascade.hh, 355
- IIRClear
 - clDSPOp, 106
- IIRDecBank
 - clRecDecimator, 184
- IIRDecimator.cc, 356
- IIRDecimator.hh, 357
- IIRFilter
 - clDSPOp, 106
- IIRInitialize
 - clDSPOp, 105

- IIRIntBank
 - clRecInterpolator, 189
- IIRInterpolator.cc, 358
- IIRInterpolator.hh, 359
- IIRMultiRate.cc, 360
- IIRMultiRate.hh, 361
- InBuf
 - clFilter, 143
 - clFIRDecimator, 145
 - clFlipBand, 153
 - clIIRDecimator, 163
- Index
 - clReBuffer, 173
- InitAbel
 - clHankel, 155
- InitCoeffsD
 - clFilter, 135
- InitCoeffsS
 - clFilter, 135
- InitDouble
 - clFilter, 137
- InitFloat
 - clFilter, 137
- InitHalves
 - clRecDecimator, 181
 - clRecInterpolator, 186
- Initialize
 - clFFTMultiRate, 131
 - clFilter, 136, 137
 - clFIRMultiRate, 149
 - clFlipBand, 152
 - clHankel, 156
 - clIIRCascade, 160
 - clIIRMultiRate, 166, 167
 - clRecDecimator, 181, 182
 - clRecInterpolator, 186, 187
 - clSemaphore, 194
- InitializeHP
 - clFilter, 138
- InitializeLP
 - clFilter, 137
- INLINE
 - DSPOp.hh, 327
- IntBuf
 - clFFTInterpolator, 129
 - clFIRInterpolator, 147
 - clIIRInterpolator, 165
 - clRecInterpolator, 189
- InternalCaller
 - clDynThreads, 119
 - clDynThreadsBase, 122
- Interpolate
 - clDSPOp, 90
 - clDSPVector, 115
- InterpolateAvg
 - clDSPOp, 90, 91
 - clDSPVector, 115
- iThreadCount
 - clDynThreadsBase, 122
- iType
 - clRecDecimator, 183
 - clRecInterpolator, 188
- Klass
 - clDynThreads, 119
 - clDynThreadsBase::_stParams, 123
- lBlockSize
 - clFlipBand, 153
- lCBlockSize
 - clFlipBand, 153
- lCount
 - clReBuffer, 174
 - clReBufferT, 179
- lDecSize
 - clRecDecimator, 183
- lFactor
 - clFFTMultiRate, 131
 - clFIRMultiRate, 149
 - clIIRMultiRate, 167
 - clRecDecimator, 183
 - clRecInterpolator, 188
- lFFTLlength
 - clDSPOp, 110
- lFFTSsize
 - clDSPVector, 117
 - clFilter, 143
 - clHankel, 158
- lFilterSize
 - clFFTMultiRate, 131
 - clRecDecimator, 183
 - clRecInterpolator, 188
- lFIRLength
 - clDSPOp, 110
- lGetIndex
 - clReBuffer, 174
 - clReBufferT, 179
- lHalfSize
 - clFilter, 143

- likely
 - Compilers.hh, [274](#)
- lIntSize
 - clRecInterpolator, [188](#)
- lNewSize
 - clFilter, [143](#)
- Lock
 - clAlloc, [18](#)
- lOldSize
 - clFilter, [143](#)
- LongCompare
 - DSPOp.cc, [277](#)
- lpDBitRevWork
 - clDSPOp, [110](#)
- lPrevDestCount
 - clDSPOp, [110](#)
- lPrevSrcCount
 - clDSPOp, [110](#)
- lpSBitRevWork
 - clDSPOp, [110](#)
- lPutIndex
 - clReBuffer, [174](#)
 - clReBufferT, [179](#)
- lSize
 - clAlloc, [21](#)
 - clHankel, [158](#)
 - clReBuffer, [174](#)
 - clReBufferT, [179](#)
- lSpectPoints
 - clFilter, [143](#)
- lStages
 - clIIRCascade, [161](#)
- lSubRounds
 - clRecDecimator, [183](#)
 - clRecInterpolator, [188](#)
- M
 - _sDPolar, [10](#)
 - _sSPolar, [12](#)
- Magnitude
 - clDSPOp, [87](#), [88](#)
 - clDSPVector, [115](#)
- main
 - Test.cc, [374](#)
- makeect
 - clTransform4, [199](#)
 - clTransform8, [208](#)
 - clTransformS, [212](#)
- makeipt
 - clTransformS, [212](#)
- makewt
 - clTransform4, [199](#)
 - clTransform8, [208](#)
 - clTransformS, [212](#)
- mapThreads
 - clDynThreadsBase, [122](#)
- Mean
 - clDSPOp, [72](#), [73](#)
 - clDSPVector, [115](#)
- Median
 - clDSPOp, [73](#)
 - clDSPVector, [115](#)
- Method
 - clDynThreads::_stParams2, [120](#)
- Method_t
 - clDynThreads, [119](#)
- MinMax
 - clDSPOp, [72](#)
 - clDSPVector, [115](#)
- Mix
 - clDSPOp, [99](#), [100](#)
 - clDSPVector, [115](#)
- ModZeroBessel
 - clDSPOp, [49](#)
- MtxBase
 - clDynThreadsBase, [122](#)
- Mul
 - clDSPOp, [54–56](#)
- Mul2
 - clDSPOp, [57](#), [58](#)
- MulAdd
 - clDSPOp, [61](#), [62](#)
 - clDSPVector, [115](#)
- MulC
 - clDSPOp, [57](#)
 - clDSPVector, [115](#)
- Multiple
 - clDSPOp, [49](#)
- Mutex.hh, [362](#)
- Negate
 - clDSPOp, [73](#), [74](#)
 - clDSPVector, [115](#)
- Normalize
 - clDSPOp, [74](#), [75](#)
 - clDSPVector, [115](#)
- Notify
 - clCondition, [23](#)
 - clPthCond, [170](#)
- NotifyAll

- clCondition, 23
- operator *
 - clDSPVector, 115
- operator *=
 - clDSPVector, 115
- operator char *
 - clAlloc, 19
- operator double *
 - clAlloc, 20
- operator float *
 - clAlloc, 20
- operator int
 - clException, 125
- operator int *
 - clAlloc, 19
- operator long *
 - clAlloc, 19
- operator long double *
 - clAlloc, 20
- operator short *
 - clAlloc, 19
- operator std::string
 - clException, 125
- operator stDCplx *
 - clDSPAlloc, 24
- operator stDPolar *
 - clDSPAlloc, 24
- operator stSCplx *
 - clDSPAlloc, 24
- operator stSPolar *
 - clDSPAlloc, 24
- operator unsigned char *
 - clAlloc, 19
- operator unsigned int *
 - clAlloc, 19
- operator unsigned long *
 - clAlloc, 19
- operator unsigned short *
 - clAlloc, 19
- operator void *
 - clAlloc, 20
- operator+
 - clDSPVector, 115
- operator+=
 - clDSPVector, 115
- operator-
 - clDSPVector, 115
- operator-=
 - clDSPVector, 115
- operator/
 - clDSPVector, 115
- operator/=
 - clDSPVector, 115
- operator=
 - clAlloc, 20
 - clReBuffer, 174
 - clReBufferT, 178
- operator[]
 - clReBufferT, 179
- OutBuf
 - clFilter, 143
 - clFIRInterpolator, 147
 - clFlipBand, 153
 - clIIRInterpolator, 165
- P
 - _sDPolar, 10
 - _sSPolar, 12
 - _uDCoord, 13
 - _uSCoord, 14
- Pack
 - clDSPOp, 101
 - clDSPVector, 115
- PeakLevel
 - clDSPOp, 92, 93
 - clDSPVector, 115
- Phase
 - clDSPOp, 88
 - clDSPVector, 115
- Polar2Cart
 - clDSPOp, 42, 43
- PolarToCart
 - clDSPOp, 84, 85
 - clDSPVector, 115
- Post
 - clSemaphore, 194
- Power
 - clDSPOp, 88
 - clDSPVector, 115
- PowerPhase
 - clDSPOp, 89
 - clDSPVector, 115
- prefetch
 - Compilers.hh, 274
- Prev
 - clFilter, 143
- Proc
 - clFilter, 143
 - clFlipBand, 153

- Process
 - clIIRCascade, [160](#), [161](#)
- Process0
 - clHankel, [156](#), [157](#)
- Process1
 - clHankel, [157](#)
- Product
 - clDSPOp, [75](#)
- pthcCond
 - clCondition, [23](#)
 - clPthCond, [170](#)
- PthCond.hh, [363](#)
- pthmMutex
 - clMutex, [169](#)
 - clPthMutex, [171](#)
- PthMutex.hh, [364](#)
- Ptr
 - clDSPVector, [115](#)
- ptrwLock
 - clRWLock, [192](#)
- Put
 - clFFTDecimator, [126](#), [127](#)
 - clFFTInterpolator, [128](#), [129](#)
 - clFilter, [139](#), [140](#)
 - clFIRDecimator, [145](#)
 - clFIRInterpolator, [147](#)
 - clFlipBand, [152](#)
 - clIIRDecimator, [163](#)
 - clIIRInterpolator, [165](#)
 - clReBuffer, [173](#)
 - clReBufferT, [177](#)
 - clRecDecimator, [182](#)
 - clRecInterpolator, [187](#)
- R
 - _sDCplx, [9](#)
 - _sSCplx, [11](#)
- RadToDeg
 - clDSPOp, [102](#)
- rdft
 - clTransform4, [200](#), [201](#)
 - clTransform8, [208](#)
 - clTransformS, [212](#)
- ReadyFilterD
 - clFilter, [136](#)
- ReadyFilterS
 - clFilter, [136](#)
- ReBuffer
 - clDSPOp, [101](#), [102](#)
- ReBuffer.cc, [365](#)
- ReBuffer.hh, [366](#)
- ReBufferT.hh, [367](#)
- RECDEC_MAX_SUB_ROUNDS
 - RecDecimator.hh, [369](#)
- RecDecimator.cc, [368](#)
- RecDecimator.hh, [369](#)
- RecDecimator.hh
 - RECDEC_MAX_SUB_ROUNDS, [369](#)
- RECINT_MAX_SUB_ROUNDS
 - RecInterpolator.hh, [371](#)
- RecInterpolator.cc, [370](#)
- RecInterpolator.hh, [371](#)
- RecInterpolator.hh
 - RECINT_MAX_SUB_ROUNDS, [371](#)
- Release
 - clMutex, [169](#)
 - clPthMutex, [171](#)
 - clRWLock, [191](#)
- Resample
 - clDSPOp, [91](#)
 - clDSPVector, [115](#)
- ResampleAvg
 - clDSPOp, [91](#)
 - clDSPVector, [115](#)
- Resize
 - clAlloc, [17](#)
 - clReBufferT, [178](#)
- Reverse
 - clDSPOp, [75](#), [76](#)
 - clDSPVector, [115](#)
- rftbsub
 - clTransform4, [199](#)
 - clTransform8, [208](#)
 - clTransformS, [212](#)
- rftfsb
 - clTransform4, [199](#)
 - clTransform8, [208](#)
 - clTransformS, [212](#)
- RMS
 - clDSPOp, [91](#), [92](#)
 - clDSPVector, [115](#)
- Round
 - clDSPOp, [50](#)
- RWLock.hh, [372](#)
- _XOPEN_SOURCE, [372](#)
- SBitRevWork
 - clDSPOp, [110](#)

- Scale
 - clDSPOp, [76, 77](#)
 - clDSPVector, [115](#)
- Scale01
 - clDSPOp, [77, 78](#)
 - clDSPVector, [115](#)
- SCosSinTable
 - clDSPOp, [110](#)
- Self
 - clDynThreadsBase, [122](#)
- Semaphore.hh, [373](#)
- semSemaphore
 - clSemaphore, [195](#)
- Set
 - clDSPOp, [64, 65](#)
 - clDSPVector, [115](#)
- SetCoeffs
 - clFilter, [138](#)
- SetSched
 - clDynThreadsBase, [122](#)
- SetSize
 - clReBufferT, [178](#)
- Size
 - clAlloc, [17](#)
 - clReBufferT, [178](#)
- Sort
 - clDSPOp, [78](#)
 - clDSPVector, [115](#)
- Spatialize
 - clDSPOp, [100](#)
- Sqrt
 - clDSPOp, [62, 63](#)
 - clDSPVector, [115](#)
- Square
 - clDSPOp, [79, 80](#)
 - clDSPVector, [115](#)
- stDCplx
 - dsptypes.h, [329](#)
- StdDev
 - clDSPOp, [78, 79](#)
 - clDSPVector, [115](#)
- stDPolar
 - dsptypes.h, [330](#)
- Stop
 - clUserThreads, [213](#)
- stParams
 - clDynThreadsBase, [122](#)
- stParams2
 - clDynThreads, [119](#)
- stpDCplx
 - dsptypes.h, [329](#)
- stpDPolar
 - dsptypes.h, [330](#)
- stpParams
 - clDynThreadsBase, [122](#)
- stpParams2
 - clDynThreads, [119](#)
- stpSCplx
 - dsptypes.h, [329](#)
- stpSPolar
 - dsptypes.h, [329](#)
- strMsg
 - clException, [125](#)
- stSCplx
 - dsptypes.h, [329](#)
- stSPolar
 - dsptypes.h, [329](#)
- Sub
 - clDSPOp, [52–54](#)
- Sum
 - clDSPOp, [79](#)
 - clDSPVector, [115](#)
- T4_INLINE
 - Transform4.hh, [376](#)
- T8_INLINE
 - Transform8.hh, [378](#)
- Test.cc, [374](#)
 - main, [374](#)
- Tfrm
 - clDSPOp, [110](#)
- Thread1
 - clUserThreads, [213](#)
- Thread2
 - clUserThreads, [213](#)
- Thread3
 - clUserThreads, [213](#)
- ThreadMap_t
 - DynThreads.hh, [333](#)
- Transform4.cc, [375](#)
- Transform4.hh, [376](#)
 - T4_INLINE, [376](#)
- Transform8.cc, [377](#)
- Transform8.hh, [378](#)
 - T8_INLINE, [378](#)
- TransformS.cc, [379](#)
- TransformS.hh, [380](#)
- TransformS.hh
 - TS_INLINE, [380](#)
- TryLock

- clMutex, 169
- TryRead
 - clRWLock, 191
- TryWait
 - clSemaphore, 194
- TryWrite
 - clRWLock, 191
- TS_INLINE
 - TransformS.hh, 380
- UninitAbe1
 - clHankel, 156
- Uninitialize
 - clFFTDecimator, 126
 - clFFTInterpolator, 128
 - clFFTMultiRate, 131
 - clFilter, 138
 - clFIRDecimator, 144
 - clFIRInterpolator, 146
 - clFIRMultiRate, 149
 - clFlipBand, 152
 - clHankel, 156
 - clIIRCascade, 160
 - clIIRDecimator, 162
 - clIIRInterpolator, 164
 - clIIRMultiRate, 167
 - clRecDecimator, 182
 - clRecInterpolator, 187
- unlikely
 - Compilers.hh, 274
- UnLock
 - clAlloc, 18
- utDCoord
 - dsptypes.h, 330
- utpDCoord
 - dsptypes.h, 330
- utpSCoord
 - dsptypes.h, 330
- utSCoord
 - dsptypes.h, 330
- Variance
 - clDSPOp, 92
 - clDSPVector, 115
- vpDTfrm
 - clDSPOp, 110
- vpParam
 - clDynThreads::_stParams2, 120
 - clDynThreadsBase::_stParams, 123
- vpPtr
 - clAlloc, 21
- vpSTfrm
 - clDSPOp, 110
- Wait
 - clCondition, 23
 - clDynThreadsBase, 122
 - clMutex, 169
 - clPthCond, 170
 - clPthMutex, 171
 - clSemaphore, 194
- WaitRead
 - clRWLock, 191
- WaitWrite
 - clRWLock, 191
- what
 - clException, 125
- WinBartlett
 - clDSPOp, 93
 - clDSPVector, 115
- WinBlackman
 - clDSPOp, 93, 94
 - clDSPVector, 115
- WinBlackmanHarris
 - clDSPOp, 94
 - clDSPVector, 115
- WinCosTapered
 - clDSPOp, 94
 - clDSPVector, 115
- WinCosTaperedA
 - clDSPOp, 95
- WinDolphChebyshev
 - clDSPOp, 98, 99
 - clDSPVector, 115
- WinExactBlackman
 - clDSPOp, 95
 - clDSPVector, 115
- WinExp
 - clDSPOp, 95, 96
 - clDSPVector, 115
- WinFlatTop
 - clDSPOp, 96
 - clDSPVector, 115
- WinGenericCos
 - clDSPOp, 96
 - clDSPVector, 115
- WinHamming
 - clDSPOp, 97
 - clDSPVector, 115

WinHanning
 clDSPOp, [97](#)
 clDSPVector, [115](#)

WinKaiser
 clDSPOp, [97](#), [98](#)
 clDSPVector, [115](#)

WinKaiserBessel
 clDSPOp, [98](#)
 clDSPVector, [115](#)

WinTukey
 clDSPOp, [98](#)
 clDSPVector, [115](#)

WrapDynThreadBase
 DynThreads.cc, [332](#)

X86-64.c, [381](#)
X86-64.h, [382](#)
X86.c, [383](#)
X86.h, [384](#)

Zero
 clDSPOp, [63](#), [64](#)
 clDSPVector, [115](#)