libDSP Reference Manual

5.0.0

Generated by Doxygen 1.3.6

Sun Nov 7 14:31:57 2004

Contents

Chapter 1

libDSP Main Page

DSP operations

Author:
Jussi Laako

Date
2004/11/07 02:00:34

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

Licensing

Chapter 2

libDSP Hierarchical Index

2.1 libDSP Class Hierarchy

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

Chapter 3

libDSP Class Index

3.1 libDSP Class List

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

Chapter 4

libDSP File Index

4.1 libDSP File List

Here is a list of all files with brief descriptions:

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.

Definition at line 42 of file dsptypes.h.

5.1.2 Member Data Documentation

5.1.2.1 double _sDCplx::R

Real.

Definition at line 44 of file dsptypes.h.

Referenced by clDSPOp::Cart2Polar(), clDSPOp::CplxAdd(), clDSPOp::CplxConj(), clDSPOp::CplxDiv(), clDSPOp::CplxExp(), clDSPOp::CplxLog(), clDSPOp::CplxLog10(), clDSPOp::CplxMul(), clDSPOp::CplxMulC(), clDSPOp::CplxRoot(), clDSPOp::CplxSub(), clDSPOp::FFTWConvert(), clFilter::GetCoeffs(), clFilter::InitCoeffsD(), clDSPOp::Polar2Cart(), clDSPOp::PowerPhase(), clHankel::Process0(), clHankel::Process1(), clDSPOp::Reverse(), and clFilter::SetCoeffs().

5.1.2.2 double _sDCplx::I

Imaginary.

Definition at line 45 of file dsptypes.h.

Referenced by clDSPOp::Cart2Polar(), clDSPOp::CplxAdd(), clDSPOp::CplxConj(), clDSPOp::CplxDiv(), clDSPOp::CplxExp(), clDSPOp::CplxLog(), clDSPOp::CplxLog10(), clDSPOp::CplxMul(), clDSPOp::CplxMulC(), clDSPOp::CplxRoot(), clDSPOp::CplxSub(), clDSPOp::FFTWConvert(), clFilter::InitCoeffsD(), clDSPOp::Polar2Cart(), clHankel::Process0(), clHankel::Process1(), clDSPOp::Reverse(), and clFilter::SetCoeffs().

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

- dsptypes.h

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.

Definition at line 60 of file dsptypes.h.

5.2.2 Member Data Documentation

5.2.2.1 double _sDPolar::M

Magnitude.

Definition at line 62 of file dsptypes.h.

Referenced by clDSPOp::Cart2Polar(), and clDSPOp::Polar2Cart().

5.2.2.2 double _sDPolar::P

Phase.

Definition at line 63 of file dsptypes.h.

Referenced by clDSPOp::Cart2Polar(), and clDSPOp::Polar2Cart().

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

- dsptypes.h

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.

Definition at line 33 of file dsptypes.h.

5.3.2 Member Data Documentation

5.3.2.1 float _sSCplx::R

Real.

Definition at line 35 of file dsptypes.h.

Referenced by clDSPOp::Cart2Polar(), clDSPOp::CplxAdd(), clDSPOp::CplxConj(), clDSPOp::CplxDiv(), clDSPOp::CplxExp(), clDSPOp::CplxLog(), clDSPOp::CplxLog10(), clDSPOp::CplxMul(), clDSPOp::CplxMulC(), clDSPOp::CplxRoot(), clDSPOp::CplxSub(), clDSPOp::FFTWConvert(), clFilter::GetCoeffs(), clFilter::InitCoeffsS(), clDSPOp::Polar2Cart(), clDSPOp::PowerPhase(), clHankel::Process0(), clHankel::Process1(), clDSPOp::Reverse(), and clFilter::SetCoeffs().

5.3.2.2 float _sSCplx::I

Imaginary.

Definition at line 36 of file dsptypes.h.

Referenced by clDSPOp::Cart2Polar(), clDSPOp::CplxAdd(), clDSPOp::CplxConj(), clDSPOp::CplxDiv(), clDSPOp::CplxExp(), clDSPOp::CplxLog(), clDSPOp::CplxLog10(), clDSPOp::CplxMul(), clDSPOp::CplxMulC(), clDSPOp::CplxRoot(), clDSPOp::CplxSub(), clDSPOp::FFTWConvert(), clFilter::InitCoeffsS(), clDSPOp::Polar2Cart(), clHankel::Process0(), clHankel::Process1(), clDSPOp::Reverse(), and clFilter::SetCoeffs().

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

- dsptypes.h

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.

Definition at line 51 of file dsptypes.h.

5.4.2 Member Data Documentation

5.4.2.1 float _sSPolar::M

Magnitude.

Definition at line 53 of file dsptypes.h.

Referenced by clDSPOp::Cart2Polar(), and clDSPOp::Polar2Cart().

5.4.2.2 float _sSPolar::P

Phase.

Definition at line 54 of file dsptypes.h.

Referenced by clDSPOp::Cart2Polar(), and clDSPOp::Polar2Cart().

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

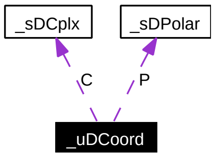
- dsptypes.h

5.5 _uDCoord Union Reference

Double precision cartesian/polar datatype.

`#include <dsptypes.h>`

Collaboration diagram for _uDCoord:



Public Attributes

- stDCplx C

  Cartesian.

- stDPolar P

  Polar.

5.5.1 Detailed Description

Double precision cartesian/polar datatype.

Definition at line 78 of file dsptypes.h.

5.5.2 Member Data Documentation

5.5.2.1 stDCplx _uDCoord::C

Cartesian.

Definition at line 80 of file dsptypes.h.

Referenced by clDSPOp::Cart2Polar(), and clDSPOp::Polar2Cart().

5.5.2.2 stDPolar _uDCoord::P

Polar.

Definition at line 81 of file dsptypes.h.

Referenced by clDSPOp::Cart2Polar(), and clDSPOp::Polar2Cart().

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

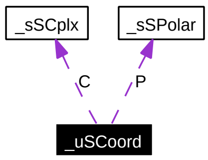
- dsptypes.h

5.6 _uSCoord Union Reference

Single precision cartesian/polar datatype.

`#include <dsptypes.h>`

Collaboration diagram for _uSCoord:



Public Attributes

- stSCplx C

  Cartesian.

- stSPolar P

  Polar.

5.6.1 Detailed Description

Single precision cartesian/polar datatype.

Definition at line 69 of file dsptypes.h.

5.6.2 Member Data Documentation

5.6.2.1 stSCplx _uSCoord::C

Cartesian.

Definition at line 71 of file dsptypes.h.

Referenced by clDSPOp::Cart2Polar(), and clDSPOp::Polar2Cart().

5.6.2.2 stSPolar _uSCoord::P

Polar.

Definition at line 72 of file dsptypes.h.

Referenced by clDSPOp::Cart2Polar(), and clDSPOp::Polar2Cart().

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

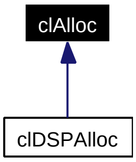
- dsptypes.h

5.7 clAlloc Class Reference

Class for memory allocation operations.

`#include <Alloc.hh>`

Inheritance diagram for clAlloc:



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.

Definition at line 48 of file Alloc.hh.

5.7.2 Constructor & Destructor Documentation

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

Constructor; initializes empty allocation.

Definition at line 58 of file Alloc.hh.

References bLocked, lSize, and vpPtr.

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

Copy constructor; copies instance.

Definition at line 67 of file Alloc.hh.

References Copy().

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

Definition at line 77 of file Alloc.hh.

References bLocked, lSize, Size(), and vpPtr.

5.7.2.4 clAlloc::~clAlloc () `[inline]`

Destructor; frees allocated memory block.

Definition at line 87 of file Alloc.hh.

References Free().

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

Definition at line 97 of file Alloc.hh.

References ALLOC_ALIGNMENT, Free(), lSize, and vpPtr.

Referenced by clAlloc(), Copy(), clDSPOp::DCTInitialize(), clDSPOp::FFTInitialize(), clDSPOp::FIRAllocate(), clDSPOp::FIRFilter(), clIIRDecimator::Get(), clFIRDecimator::Get(), clFFTDecimator::Get(), clReBuffer::GetPtr(), clHankel::InitAbel(), clRecInterpolator::Initialize(), clRecDecimator::Initialize(), clHankel::Initialize(), clFlipBand::Initialize(), clFilter::Initialize(), clIIRInterpolator::Put(), clFIRInterpolator::Put(), clFFTInterpolator::Put(), and clRingBuffer< TRingBuffer_t >::SetSize().

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

Definition at line 124 of file Alloc.hh.

References ALLOC_ALIGNMENT, bLocked, Free(), lSize, UnLock(), and vpPtr.

Referenced by clReBufferT< TDSPVector_t >::CheckSize(), and clReBuffer::CheckSize().

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.

Definition at line 161 of file Alloc.hh.

References bLocked, lSize, UnLock(), and vpPtr.

Referenced by clReBufferT< TDSPVector_t >::Clear(), clReBuffer::Clear(), Copy(), clDSPOp::DCTUninitialize(), clDSPOp::FFTUninitialize(), clDSPOp::FIRFree(), Resize(), Size(), clHankel::UninitAbel(), clRecInterpolator::Uninitialize(), clRecDecimator::Uninitialize(), clIIRInterpolator::Uninitialize(), clIIRDecimator::Uninitialize(), clHankel::Uninitialize(), clFlipBand::Uninitialize(), clFIRInterpolator::Uninitialize(), clFIRDecimator::Uninitialize(), clFilter::Uninitialize(), clFFTInterpolator::Uninitialize(), clFFTDecimator::Uninitialize(), and ~clAlloc().

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

Get pointer to allocated block.

Returns:
Pointer to memory block

Definition at line 180 of file Alloc.hh.

References vpPtr.

Referenced by clReBufferT< TDSPVector_t >::CopyGet(), and clHankel::DoAbel().

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

Get size of allocation.

Returns:
Size, in bytes, of allocation

Definition at line 187 of file Alloc.hh.

References lSize.

Referenced by clReBufferT< TDSPVector_t >::CheckSize(), and clReBuffer::CheckSize().

5.7.3.6 void clAlloc::Lock () `[inline]`

Lock memory block (prevent paging out).

Definition at line 192 of file Alloc.hh.

References bLocked, lSize, and vpPtr.

Referenced by clRingBuffer< TRingBuffer_t >::Lock().

5.7.3.7 void clAlloc::UnLock () `[inline]`

Unlock memory block (allow paging out).

Definition at line 204 of file Alloc.hh.

References bLocked, lSize, and vpPtr.

Referenced by Free(), Resize(), and clRingBuffer< TRingBuffer_t >::UnLock().

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

Definition at line 219 of file Alloc.hh.

References Free(), lSize, Size(), and vpPtr.

Referenced by clAlloc(), CopyTo(), GetCopy(), clReBuffer::operator=(), and operator=().

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

Definition at line 231 of file Alloc.hh.

References Copy().

5.7.3.10 clAlloc clAlloc::GetCopy () `[inline]`

Return copy of this instance.

Returns:

Copy of this memory block

Definition at line 240 of file Alloc.hh.

References Copy().

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

Return pointer to memory block.

Definition at line 250 of file Alloc.hh.

References vpPtr.

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.

Definition at line 253 of file Alloc.hh.

References vpPtr.

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.

Definition at line 256 of file Alloc.hh.

References vpPtr.

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.

Definition at line 259 of file Alloc.hh.

References vpPtr.

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.

Definition at line 262 of file Alloc.hh.

References vpPtr.

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.

Definition at line 265 of file Alloc.hh.

References vpPtr.

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.

Definition at line 268 of file Alloc.hh.

References vpPtr.

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.

Definition at line 271 of file Alloc.hh.

References vpPtr.

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.

Definition at line 274 of file Alloc.hh.

References vpPtr.

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.

Definition at line 277 of file Alloc.hh.

References vpPtr.

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.

Definition at line 280 of file Alloc.hh.

References vpPtr.

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.

Definition at line 283 of file Alloc.hh.

References vpPtr.

5.7.3.23 clAlloc::operator void $*$ () [inline]

Definition at line 285 of file Alloc.hh.

References vpPtr.

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

Definition at line 294 of file Alloc.hh.

References Copy().

5.7.4 Member Data Documentation

5.7.4.1 bool clAlloc::bLocked [protected]

Definition at line 51 of file Alloc.hh.

Referenced by clAlloc(), Free(), Lock(), Resize(), and UnLock().

5.7.4.2 long clAlloc::lSize [protected]

Definition at line 52 of file Alloc.hh.

Referenced by clAlloc(), Copy(), Free(), GetSize(), Lock(), Resize(), Size(), and UnLock().

5.7.4.3 void$*$ clAlloc::vpPtr [protected]

Definition at line 53 of file Alloc.hh.

Referenced by clAlloc(), Copy(), Free(), GetPtr(), Lock(), operator char $*$(), operator double $*$(), operator float $*$(), operator int $*$(), operator long $*$(), operator long double

$*$(), operator short $*$(), operator unsigned char $*$(), operator unsigned int $*$(), operator unsigned long $*$(), operator unsigned short $*$(), operator void $*$(), Resize(), Size(), and UnLock().

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

- Alloc.hh

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.

Definition at line 40 of file Condition.hh.

5.8.2 Constructor & Destructor Documentation

5.8.2.1 clCondition::clCondition () `[inline]`

Constructor; creates and initializes the condition variable.

Definition at line 47 of file Condition.hh.

References pthcCond.

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

Destructor; destroys the condition variable.

Definition at line 51 of file Condition.hh.

References pthcCond.

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

Definition at line 61 of file Condition.hh.

References pthcCond.

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 varibale

iCondTO Timeout in milliseconds

Returns:
Success

Definition at line 79 of file Condition.hh.

References pthcCond.

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.

Definition at line 102 of file Condition.hh.

References pthcCond.

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.

Definition at line 109 of file Condition.hh.

References pthcCond.

5.8.4 Member Data Documentation

5.8.4.1 pthread_cond_t clCondition::pthcCond [private]

Definition at line 42 of file Condition.hh.

Referenced by clCondition(), Notify(), NotifyAll(), Wait(), and ~clCondition().

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

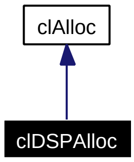
- Condition.hh

5.9 clDSPAlloc Class Reference

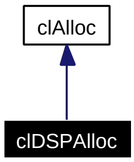
Class specialization to support automatic typecasts to cartesian/polar datatypes.

`#include <DSPOp.hh>`

Inheritance diagram for clDSPAlloc:



Collaboration diagram for clDSPAlloc:



Public Member Functions

- clDSPAlloc ()
- clDSPAlloc (const clDSPAlloc &CopySrc)
- clDSPAlloc (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.

Definition at line 124 of file DSPOp.hh.

5.9.2 Constructor & Destructor Documentation

5.9.2.1 clDSPAlloc::clDSPAlloc () `[inline]`

Definition at line 127 of file DSPOp.hh.

5.9.2.2 clDSPAlloc::clDSPAlloc (const clDSPAlloc & *CopySrc*) `[inline]`

Definition at line 128 of file DSPOp.hh.

5.9.2.3 clDSPAlloc::clDSPAlloc (long *lAllocSize*) `[inline]`

Definition at line 129 of file DSPOp.hh.

5.9.3 Member Function Documentation

5.9.3.1 clDSPAlloc::operator stSCplx * () `[inline]`

Definition at line 130 of file DSPOp.hh.

References stpSCplx.

5.9.3.2 clDSPAlloc::operator stDCplx * () `[inline]`

Definition at line 132 of file DSPOp.hh.

References stpDCplx.

5.9.3.3 clDSPAlloc::operator stSPolar * () `[inline]`

Definition at line 134 of file DSPOp.hh.

References stpSPolar.

5.9.3.4 clDSPAlloc::operator stDPolar * () `[inline]`

Definition at line 136 of file DSPOp.hh.

References stpDPolar.

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

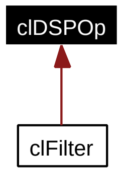
- DSPOp.hh

5.10 clDSPOp Class Reference

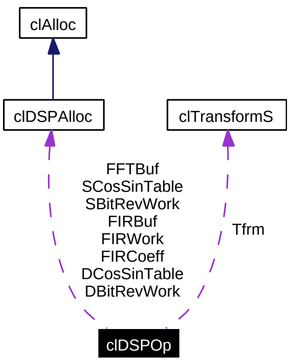
Class of DSP operations.

`#include <DSPOp.hh>`

Inheritance diagram for clDSPOp:



Collaboration diagram for clDSPOp:



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 inversing 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.

- void FFTUninitialize ()

    Uninitialize FFT.

- void FFTi (stpSCplx, float ∗)

    FFT half in-place, source vector is modified.

- void FFTi (stpDCplx, double ∗)
- void FFTo (stpSCplx, const float ∗)

    FFT out-of-place.

- void FFTo (stpDCplx, const double ∗)
- void FFTo (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)
- void DCTInitialize (long)

    DCT initialize.

- void DCTUninitialize ()

    Uninitialize DCT.

- void DCTi (float ∗)

    DCT in-place.

- void DCTi (double ∗)
- void DCTo (float ∗, const float ∗)

    DCT out-of-place.

- void DCTo (double ∗, const double ∗)
- void IDCTi (float ∗)

  Inverse DCT in-place.

- void IDCTi (double ∗)
- void IDCTo (float ∗, const float ∗)

  Inverse DCT out-of-place.

- void IDCTo (double ∗, const double ∗)

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 squre-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)

  Iterpolate 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 lFFTLength
- float fFFTScale
- double dFFTScale

- long ∗ lpSBitRevWork
- long ∗ lpDBitRevWork
- float ∗ fpCosSinTable
- double ∗ dpCosSinTable
- void ∗ vpSTfrm
- void ∗ vpDTfrm
- clDSPAlloc SBitRevWork
- clDSPAlloc DBitRevWork
- clDSPAlloc SCosSinTable
- clDSPAlloc DCosSinTable
- clDSPAlloc FFTBuf
- clTransformS Tfrm

5.10.1 Detailed Description

Class of DSP operations.

Definition at line 149 of file DSPOp.hh.

5.10.2 Constructor & Destructor Documentation

5.10.2.1 clDSPOp::clDSPOp ()

Definition at line 769 of file DSPOp.cc.

References bFFTInitialized, dpCosSinTable, dPI, fpCosSinTable, fPI, lpDBitRevWork, lPrevDestCount, lPrevSrcCount, and lpSBitRevWork.

5.10.2.2 clDSPOp::∼clDSPOp ()

Definition at line 800 of file DSPOp.cc.

References bFFTInitialized, and FFTUninitialize().

5.10.3 Member Function Documentation

5.10.3.1 INLINE void clDSPOp::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

Definition at line 146 of file DSPOp.cc.

References INLINE.

Referenced by CartToPolar(), CplxLog(), and CplxLog10().

5.10.3.2 INLINE void clDSPOp::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.

Definition at line 158 of file DSPOp.cc.

References INLINE.

5.10.3.3 INLINE void clDSPOp::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

Definition at line 170 of file DSPOp.cc.

References _sSCplx::I, INLINE, _sSCplx::R, and stpSCplx.

5.10.3.4 INLINE void clDSPOp::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.

Definition at line 182 of file DSPOp.cc.

References _sDCplx::I, INLINE, _sDCplx::R, and stpDCplx.

5.10.3.5 INLINE void clDSPOp::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

Definition at line 194 of file DSPOp.cc.

References _sSCplx::I, INLINE, _sSPolar::M, _sSPolar::P, _sSCplx::R, stpSCplx, and stpSPolar.

5.10.3.6 INLINE void clDSPOp::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.

Definition at line 205 of file DSPOp.cc.

References _sDCplx::I, INLINE, _sDPolar::M, _sDPolar::P, _sDCplx::R, stpDCplx, and stpDPolar.

5.10.3.7 INLINE void clDSPOp::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

Definition at line 216 of file DSPOp.cc.

References _uSCoord::C, _sSCplx::I, INLINE, _sSPolar::M, _sSPolar::P, _uSCoord::P, _sSCplx::R, and utpSCoord.

5.10.3.8 INLINE void clDSPOp::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.

Definition at line 228 of file DSPOp.cc.

References _uDCoord::C, _sDCplx::I, INLINE, _sDPolar::M, _sDPolar::P, _uDCoord::P, _sDCplx::R, and utpDCoord.

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

Polar to cartesian conversion.

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

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

Note:
See Cart2Polar for details

Definition at line 240 of file DSPOp.cc.

References INLINE.

Referenced by PolarToCart().

5.10.3.10 INLINE void clDSPOp::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.

Definition at line 254 of file DSPOp.cc.

References INLINE.

5.10.3.11 INLINE void clDSPOp::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.

Definition at line 268 of file DSPOp.cc.

References _sSCplx::I, INLINE, _sSCplx::R, and stpSCplx.

5.10.3.12 INLINE void clDSPOp::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.

Definition at line 281 of file DSPOp.cc.

References _sDCplx::I, INLINE, _sDCplx::R, and stpDCplx.

5.10.3.13 INLINE void clDSPOp::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.

Definition at line 294 of file DSPOp.cc.

References _sSCplx::I, INLINE, _sSPolar::M, _sSPolar::P, _sSCplx::R, stpSCplx, and stpSPolar.

5.10.3.14 INLINE void clDSPOp::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.

Definition at line 307 of file DSPOp.cc.

References _sDCplx::I, INLINE, _sDPolar::M, _sDPolar::P, _sDCplx::R, stpDCplx, and stpDPolar.

5.10.3.15 INLINE void clDSPOp::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.

Definition at line 320 of file DSPOp.cc.

References _uSCoord::C, _sSCplx::I, INLINE, _sSPolar::M, _sSPolar::P, _uSCoord::P, _sSCplx::R, and utpSCoord.

5.10.3.16 INLINE void clDSPOp::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.

Definition at line 336 of file DSPOp.cc.

References _uDCoord::C, _sDCplx::I, INLINE, _sDPolar::M, _sDPolar::P, _uDCoord::P, _sDCplx::R, and utpDCoord.

5.10.3.17 INLINE void clDSPOp::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

Definition at line 352 of file DSPOp.cc.

References _sSCplx::I, INLINE, _sSCplx::R, and stpSCplx.

Referenced by Add().

5.10.3.18 INLINE void clDSPOp::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.

Definition at line 359 of file DSPOp.cc.

References _sDCplx::I, INLINE, _sDCplx::R, and stpDCplx.

5.10.3.19 INLINE void clDSPOp::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

Definition at line 366 of file DSPOp.cc.

References _sSCplx::I, INLINE, _sSCplx::R, and stpSCplx.

5.10.3.20 INLINE void clDSPOp::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.

Definition at line 374 of file DSPOp.cc.

References _sDCplx::I, INLINE, _sDCplx::R, and stpDCplx.

5.10.3.21 INLINE void clDSPOp::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

Definition at line 382 of file DSPOp.cc.

References _sSCplx::I, INLINE, _sSCplx::R, and stpSCplx.

Referenced by Sub().

5.10.3.22 INLINE void clDSPOp::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.

Definition at line 389 of file DSPOp.cc.

References _sDCplx::I, INLINE, _sDCplx::R, and stpDCplx.

5.10.3.23 INLINE void clDSPOp::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

Definition at line 396 of file DSPOp.cc.

References _sSCplx::I, INLINE, _sSCplx::R, and stpSCplx.

5.10.3.24 INLINE void clDSPOp::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.

Definition at line 404 of file DSPOp.cc.

References _sDCplx::I, INLINE, _sDCplx::R, and stpDCplx.

5.10.3.25 INLINE void clDSPOp::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

Definition at line 412 of file DSPOp.cc.

References _sSCplx::I, INLINE, _sSCplx::R, and stpSCplx.

Referenced by CplxPow(), and Mul().

5.10.3.26 INLINE void clDSPOp::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.

Definition at line 419 of file DSPOp.cc.

References _sDCplx::I, INLINE, _sDCplx::R, and stpDCplx.

5.10.3.27 INLINE void clDSPOp::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

Definition at line 426 of file DSPOp.cc.

References _sSCplx::I, INLINE, _sSCplx::R, and stpSCplx.

5.10.3.28 INLINE void clDSPOp::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.

Definition at line 438 of file DSPOp.cc.

References _sDCplx::I, INLINE, _sDCplx::R, and stpDCplx.

5.10.3.29 INLINE void clDSPOp::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

Definition at line 450 of file DSPOp.cc.

References _sSCplx::I, INLINE, _sSCplx::R, and stpSCplx.

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.

Definition at line 460 of file DSPOp.cc.

References _sDCplx::I, INLINE, _sDCplx::R, and stpDCplx.

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

Definition at line 470 of file DSPOp.cc.

References _sSCplx::I, INLINE, _sSCplx::R, and stpSCplx.

Referenced by MulC().

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.

Definition at line 482 of file DSPOp.cc.

References _sDCplx::I, INLINE, _sDCplx::R, and stpDCplx.

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

Definition at line 494 of file DSPOp.cc.

References _sSCplx::I, INLINE, _sSCplx::R, and stpSCplx.

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.

Definition at line 504 of file DSPOp.cc.

References _sDCplx::I, INLINE, _sDCplx::R, and stpDCplx.

5.10.3.35 INLINE void clDSPOp::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

Definition at line 514 of file DSPOp.cc.

References _sSCplx::I, INLINE, _sSCplx::R, and stpSCplx.

Referenced by CplxRoot(), and Div().

5.10.3.36 INLINE void clDSPOp::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.

Definition at line 528 of file DSPOp.cc.

References _sDCplx::I, INLINE, _sDCplx::R, and stpDCplx.

5.10.3.37 INLINE void clDSPOp::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

Definition at line 542 of file DSPOp.cc.

References _sSCplx::I, INLINE, _sSCplx::R, and stpSCplx.

5.10.3.38 INLINE void clDSPOp::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.

Definition at line 554 of file DSPOp.cc.

References _sDCplx::I, INLINE, _sDCplx::R, and stpDCplx.

5.10.3.39 INLINE void clDSPOp::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

Definition at line 566 of file DSPOp.cc.

References _sSCplx::I, INLINE, _sSCplx::R, and stpSCplx.

Referenced by CplxPow().

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

Definition at line 582 of file DSPOp.cc.

References _sDCplx::I, INLINE, _sDCplx::R, and stpDCplx.

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

Complex natural logarithm.

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

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

Definition at line 598 of file DSPOp.cc.

References Cart2Polar(), _sSCplx::I, INLINE, _sSCplx::R, and stpSCplx.

Referenced by CplxPow().

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

Definition at line 605 of file DSPOp.cc.

References Cart2Polar(), _sDCplx::I, INLINE, _sDCplx::R, and stpDCplx.

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

Complex 10-base logarithm.

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

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

Definition at line 612 of file DSPOp.cc.

References Cart2Polar(), _sSCplx::I, INLINE, _sSCplx::R, and stpSCplx.

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

Definition at line 619 of file DSPOp.cc.

References Cart2Polar(), _sDCplx::I, INLINE, _sDCplx::R, and stpDCplx.

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

Complex x raised to y.

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

Definition at line 626 of file DSPOp.cc.

References CplxExp(), CplxLog(), CplxMul(), INLINE, stpSCplx, and stSCplx.

Referenced by CplxRoot().

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

Definition at line 637 of file DSPOp.cc.

References CplxExp(), CplxLog(), CplxMul(), INLINE, stDCplx, and stpDCplx.

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

Complex root y of x.

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

Definition at line 648 of file DSPOp.cc.

References CplxDiv(), CplxPow(), _sSCplx::I, INLINE, _sSCplx::R, stpSCplx, and stSCplx.

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

Definition at line 660 of file DSPOp.cc.

References CplxDiv(), CplxPow(), _sDCplx::I, INLINE, _sDCplx::R, stDCplx, and stpDCplx.

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

Definition at line 417 of file DSPOp.hh.

References _sSCplx::I, and stpSCplx.

Referenced by Reverse().

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

Definition at line 419 of file DSPOp.hh.

References _sDCplx::I, and stpDCplx.

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

Definition at line 684 of file DSPOp.cc.

References _sSCplx::I, INLINE, _sSCplx::R, and stpSCplx.

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

Definition at line 691 of file DSPOp.cc.

References _sDCplx::I, INLINE, _sDCplx::R, and stpDCplx.

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

Return multiple of n.

$$!n$$

Parameters:

Value Value of n

Definition at line 698 of file DSPOp.cc.

References INLINE.

Referenced by ModZeroBessel().

5.10.3.54 INLINE float clDSPOp::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

Definition at line 711 of file DSPOp.cc.

References DSP_MAXBESSEL, INLINE, and Multiple().

Referenced by WinKaiser(), and WinKaiserBessel().

5.10.3.55 INLINE double clDSPOp::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.

Definition at line 727 of file DSPOp.cc.

References DSP_MAXBESSEL, INLINE, and Multiple().

5.10.3.56 float clDSPOp::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

Definition at line 743 of file DSPOp.cc.

Referenced by WinDolphChebyshev().

5.10.3.57 double clDSPOp::ChebyshevPolynom (double, double) `[static, protected]`

Definition at line 756 of file DSPOp.cc.

5.10.3.58 signed long clDSPOp::Round (float) `[static]`

Round floatingpoint number to integer.

Definition at line 112 of file DSPOp.cc.

Referenced by Convert(), dsp_round(), dsp_roundf(), WinCosTapered(), and WinCosTaperedA().

5.10.3.59 signed long clDSPOp::Round (double) [static]

Definition at line 129 of file DSPOp.cc.

5.10.3.60 void clDSPOp::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

Definition at line 809 of file DSPOp.cc.

Referenced by dsp_add(), dsp_add2(), dsp_add2f(), dsp_add3(), dsp_add3f(), dsp_addf(), dsp_cadd(), dsp_cadd2(), dsp_cadd2f(), dsp_cadd3(), dsp_cadd3f(), dsp_caddf(), clDSPVector< TDSPVector_t >::operator+(), and clDSPVector< TDSPVector_t >::operator+=().

5.10.3.61 void clDSPOp::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.

Definition at line 837 of file DSPOp.cc.

5.10.3.62 void clDSPOp::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.

Definition at line 861 of file DSPOp.cc.

References CplxAdd(), stpSCplx, and stSCplx.

5.10.3.63 void clDSPOp::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.

Definition at line 876 of file DSPOp.cc.

References CplxAdd(), stDCplx, and stpDCplx.

5.10.3.64 void clDSPOp::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

Definition at line 891 of file DSPOp.cc.

5.10.3.65 void clDSPOp::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.

Definition at line 919 of file DSPOp.cc.

5.10.3.66 void clDSPOp::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.

Definition at line 943 of file DSPOp.cc.

References CplxAdd(), and stpSCplx.

5.10.3.67 void clDSPOp::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.

Definition at line 958 of file DSPOp.cc.

References CplxAdd(), and stpDCplx.

5.10.3.68 void clDSPOp::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

Definition at line 973 of file DSPOp.cc.

5.10.3.69 void clDSPOp::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.

Definition at line 1002 of file DSPOp.cc.

5.10.3.70 void clDSPOp::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.

Definition at line 1027 of file DSPOp.cc.

References CplxAdd(), and stpSCplx.

5.10.3.71 void clDSPOp::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.

Definition at line 1045 of file DSPOp.cc.

References CplxAdd(), and stpDCplx.

5.10.3.72 void clDSPOp::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

Definition at line 1063 of file DSPOp.cc.

Referenced by dsp_csub(), dsp_csub2(), dsp_csub2f(), dsp_csub3(), dsp_csub3f(), dsp_csubf(), dsp_sub(), dsp_sub2(), dsp_sub2f(), dsp_sub3(), dsp_sub3f(), dsp_subf(), clDSPVector< TDSPVector_t >::operator-(), and clDSPVector< TDSPVector_-t >::operator-=().

5.10.3.73 void clDSPOp::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.

Definition at line 1078 of file DSPOp.cc.

5.10.3.74 void clDSPOp::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.

Definition at line 1093 of file DSPOp.cc.

References CplxSub(), stpSCplx, and stSCplx.

5.10.3.75 void clDSPOp::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.

Definition at line 1108 of file DSPOp.cc.

References CplxSub(), stDCplx, and stpDCplx.

5.10.3.76 void clDSPOp::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

Definition at line 1123 of file DSPOp.cc.

5.10.3.77 void clDSPOp::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.

Definition at line 1138 of file DSPOp.cc.

5.10.3.78 void clDSPOp::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.

Definition at line 1153 of file DSPOp.cc.

References CplxSub(), and stpSCplx.

5.10.3.79 void clDSPOp::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.

Definition at line 1168 of file DSPOp.cc.

References CplxSub(), and stpDCplx.

5.10.3.80 void clDSPOp::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

Definition at line 1183 of file DSPOp.cc.

5.10.3.81 void clDSPOp::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.

Definition at line 1199 of file DSPOp.cc.

5.10.3.82 void clDSPOp::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.

Definition at line 1215 of file DSPOp.cc.

References CplxSub(), and stpSCplx.

5.10.3.83 void clDSPOp::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.

Definition at line 1233 of file DSPOp.cc.

References CplxSub(), and stpDCplx.

5.10.3.84 void clDSPOp::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

Definition at line 1252 of file DSPOp.cc.

Referenced by dsp_chmul(), dsp_chmul2(), dsp_chmul2f(), dsp_chmulf(), dsp_cmul(), dsp_cmul2(), dsp_cmul2f(), dsp_cmul3(), dsp_cmul3f(), dsp_cmulf(), dsp_mul(), dsp_mul2(), dsp_mul2f(), dsp_mul3(), dsp_mul3f(), dsp_mul_nip(), dsp_mulf(), dsp_-mulf_nip(), FFTi(), FFTo(), clFIRDecimator::Get(), IDCTi(), IDCTo(), clDSPVector< TDSPVector_t >::operator *(), clDSPVector< TDSPVector_t >::operator *=(), cl-FIRInterpolator::Put(), clFilter::Put(), clFilter::ReadyFilterD(), clFilter::ReadyFilter-S(), and WinDolphChebyshev().

5.10.3.85 void clDSPOp::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.

Definition at line 1280 of file DSPOp.cc.

5.10.3.86 void clDSPOp::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.

Definition at line 1304 of file DSPOp.cc.

References CplxMul(), and stpSCplx.

5.10.3.87 void clDSPOp::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.

Definition at line 1315 of file DSPOp.cc.

References CplxMul(), and stpDCplx.

5.10.3.88 void clDSPOp::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.

Definition at line 1326 of file DSPOp.cc.

References CplxMul(), stpSCplx, and stSCplx.

5.10.3.89 void clDSPOp::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.

Definition at line 1356 of file DSPOp.cc.

References CplxMul(), stDCplx, and stpDCplx.

5.10.3.90 void clDSPOp::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

Definition at line 1381 of file DSPOp.cc.

5.10.3.91 void clDSPOp::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.

Definition at line 1410 of file DSPOp.cc.

5.10.3.92 void clDSPOp::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

Definition at line 1435 of file DSPOp.cc.

5.10.3.93 void clDSPOp::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.

Definition at line 1463 of file DSPOp.cc.

5.10.3.94 void clDSPOp::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.

Definition at line 1485 of file DSPOp.cc.

References CplxMul(), and stpSCplx.

5.10.3.95 void clDSPOp::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.

Definition at line 1496 of file DSPOp.cc.

References CplxMul(), and stpDCplx.

5.10.3.96 void clDSPOp::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.

Definition at line 1507 of file DSPOp.cc.

References CplxMul(), and stpSCplx.

5.10.3.97 void clDSPOp::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.

Definition at line 1537 of file DSPOp.cc.

References CplxMul(), and stpDCplx.

5.10.3.98 void clDSPOp::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

Definition at line 1562 of file DSPOp.cc.

5.10.3.99 void clDSPOp::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.

Definition at line 1591 of file DSPOp.cc.

5.10.3.100 void clDSPOp::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.

Definition at line 1616 of file DSPOp.cc.

References CplxMul(), and stpSCplx.

5.10.3.101 void clDSPOp::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.

Definition at line 1653 of file DSPOp.cc.

References CplxMul(), and stpDCplx.

5.10.3.102 void clDSPOp::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

Definition at line 1683 of file DSPOp.cc.

References CplxMulC(), and stpSCplx.

Referenced by dsp_ccmul(), dsp_ccmul_nip(), dsp_ccmulf(), dsp_ccmulf_nip(), and clDSPVector< TDSPVector_t >::MulC().

5.10.3.103 void clDSPOp::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.

Definition at line 1694 of file DSPOp.cc.

References CplxMulC(), and stpDCplx.

5.10.3.104 void clDSPOp::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.

Definition at line 1705 of file DSPOp.cc.

References CplxMulC(), and stpSCplx.

5.10.3.105 void clDSPOp::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.

Definition at line 1718 of file DSPOp.cc.

References CplxMulC(), and stpDCplx.

5.10.3.106 void clDSPOp::Mul2 (float ∗, float ∗, const float ∗, long) [static]

Multiply two vectors in-place with third vector.

$$\left| \begin{array}{l} x(i) = x(i)z(i) \\ y(i) = y(i)z(i) \end{array} \right. ,0 \leq i \leq N-1$$

Parameters:

Dest1 Source & destination 1, x()

Dest2 Source & destination 2, y()

Src Source, z()

Count Vector length, N

Definition at line 1731 of file DSPOp.cc.

5.10.3.107 void clDSPOp::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.

Definition at line 1744 of file DSPOp.cc.

5.10.3.108 void clDSPOp::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

Definition at line 1757 of file DSPOp.cc.

5.10.3.109 void clDSPOp::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.

Definition at line 1770 of file DSPOp.cc.

5.10.3.110 void clDSPOp::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

Definition at line 1783 of file DSPOp.cc.

Referenced by dsp_cdiv(), dsp_cdiv2(), dsp_cdiv2f(), dsp_cdiv3(), dsp_cdiv3f(), dsp_cdivf(), dsp_div(), dsp_div2(), dsp_div2f(), dsp_div3(), dsp_div3f(), dsp_divf(), clDSPVector< TDSPVector_t >::operator/(), and clDSPVector< TDSPVector_-t >::operator/=().

5.10.3.111 void clDSPOp::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.

Definition at line 1798 of file DSPOp.cc.

5.10.3.112 void clDSPOp::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.

Definition at line 1813 of file DSPOp.cc.

References CplxDiv(), stpSCplx, and stSCplx.

5.10.3.113 void clDSPOp::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.

Definition at line 1828 of file DSPOp.cc.

References CplxDiv(), stDCplx, and stpDCplx.

5.10.3.114 void clDSPOp::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

Definition at line 1843 of file DSPOp.cc.

5.10.3.115 void clDSPOp::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.

Definition at line 1858 of file DSPOp.cc.

5.10.3.116 void clDSPOp::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.

Definition at line 1873 of file DSPOp.cc.

References CplxDiv(), and stpSCplx.

5.10.3.117 void clDSPOp::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.

Definition at line 1888 of file DSPOp.cc.

References CplxDiv(), and stpDCplx.

5.10.3.118 void clDSPOp::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

Definition at line 1903 of file DSPOp.cc.

5.10.3.119 void clDSPOp::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.

Definition at line 1919 of file DSPOp.cc.

5.10.3.120 void clDSPOp::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.

Definition at line 1935 of file DSPOp.cc.

References CplxDiv(), and stpSCplx.

5.10.3.121 void clDSPOp::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.

Definition at line 1953 of file DSPOp.cc.

References CplxDiv(), and stpDCplx.

5.10.3.122 void clDSPOp::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

Definition at line 1971 of file DSPOp.cc.

Referenced by clDSPVector< TDSPVector_t >::Div1x(), dsp_div1x(), dsp_div1x_nip(), dsp_div1xf(), and dsp_div1xf_nip().

5.10.3.123 void clDSPOp::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.

Definition at line 1982 of file DSPOp.cc.

5.10.3.124 void clDSPOp::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

Definition at line 1993 of file DSPOp.cc.

5.10.3.125 void clDSPOp::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.

Definition at line 2004 of file DSPOp.cc.

5.10.3.126 void clDSPOp::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

Definition at line 2015 of file DSPOp.cc.

Referenced by dsp_muladd(), dsp_muladd_nip(), dsp_muladdf(), dsp_muladdf_nip(), and clDSPVector< TDSPVector_t >::MulAdd().

5.10.3.127 void clDSPOp::MulAdd (double $*$, double, double, long) [static]

Definition at line 2039 of file DSPOp.cc.

5.10.3.128 void clDSPOp::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

Definition at line 2059 of file DSPOp.cc.

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

Definition at line 2084 of file DSPOp.cc.

5.10.3.130 void clDSPOp::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

Definition at line 2105 of file DSPOp.cc.

Referenced by clDSPVector< TDSPVector_t >::Abs(), dsp_abs(), dsp_abs_nip(), dsp_absf(), and dsp_absf_nip().

5.10.3.131 void clDSPOp::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.

Definition at line 2120 of file DSPOp.cc.

5.10.3.132 void clDSPOp::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

Definition at line 2135 of file DSPOp.cc.

5.10.3.133 void clDSPOp::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.

Definition at line 2150 of file DSPOp.cc.

5.10.3.134 void clDSPOp::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

Definition at line 2165 of file DSPOp.cc.

Referenced by dsp_sqrt(), dsp_sqrt_nip(), dsp_sqrtf(), dsp_sqrtf_nip(), and clDSPVector< TDSPVector_t >::Sqrt().

5.10.3.135 void clDSPOp::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.

Definition at line 2180 of file DSPOp.cc.

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

Calculate squre-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

Definition at line 2195 of file DSPOp.cc.

5.10.3.137 void clDSPOp::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.

Definition at line 2210 of file DSPOp.cc.

5.10.3.138 void clDSPOp::Zero (float ∗, long) [static]

Set all vector elements to zero.

Parameters:

Vect Vector

Count Vector length

Definition at line 2225 of file DSPOp.cc.

Referenced by clFilter::DesignBP(), clFilter::DesignBR(), clFilter::DesignHP(), clFilter::DesignLP(), dsp_czero(), dsp_czerof(), dsp_zero(), dsp_zerof(), FIRAllocate(), IIRClear(), clFilter::Initialize(), and clDSPVector< TDSPVector_t >::Zero().

5.10.3.139 void clDSPOp::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.

Definition at line 2240 of file DSPOp.cc.

5.10.3.140 void clDSPOp::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.

Definition at line 2255 of file DSPOp.cc.

References stpSCplx.

5.10.3.141 void clDSPOp::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.

Definition at line 2271 of file DSPOp.cc.

References stpDCplx.

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

Set all vector elements to specified value.

Parameters:

Dest Destination

Src Source value

Count Vector length

Definition at line 2287 of file DSPOp.cc.

Referenced by clFilter::DesignBP(), clFilter::DesignBR(), clFilter::DesignHP(), clFilter::DesignLP(), dsp_cset(), dsp_cset2(), dsp_cset2f(), dsp_csetf(), dsp_set(), dsp_set2(), dsp_set2f(), dsp_setf(), clFilter::Initialize(), and clDSPVector< TDSPVector_t >::Set().

5.10.3.143 void clDSPOp::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.

Definition at line 2302 of file DSPOp.cc.

5.10.3.144 void clDSPOp::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.

Definition at line 2317 of file DSPOp.cc.

References stpSCplx, and stSCplx.

5.10.3.145 void clDSPOp::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.

Definition at line 2333 of file DSPOp.cc.

References stDCplx, and stpDCplx.

5.10.3.146 void clDSPOp::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

Definition at line 2349 of file DSPOp.cc.

5.10.3.147 void clDSPOp::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.

Definition at line 2368 of file DSPOp.cc.

5.10.3.148 void clDSPOp::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.

Definition at line 2387 of file DSPOp.cc.

References stpSCplx, and stSCplx.

5.10.3.149 void clDSPOp::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.

Definition at line 2407 of file DSPOp.cc.

References stDCplx, and stpDCplx.

5.10.3.150 void clDSPOp::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

Definition at line 2427 of file DSPOp.cc.

Referenced by clDSPVector< TDSPVector_t >::Clip(), dsp_clip(), dsp_clip2(), dsp_-clip2_nip(), dsp_clip2f(), dsp_clip2f_nip(), dsp_clip_nip(), dsp_clipf(), and dsp_-clipf_nip().

5.10.3.151 void clDSPOp::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.

Definition at line 2452 of file DSPOp.cc.

5.10.3.152 void clDSPOp::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

Definition at line 2477 of file DSPOp.cc.

5.10.3.153 void clDSPOp::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.

Definition at line 2499 of file DSPOp.cc.

5.10.3.154 void clDSPOp::Clip (float $*$, float, float, long) `[static]`

Clip vector in-place to fit between specified values.

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

Parameters:

Vect Source & destination, x()

Min Minimum value, a

Max Maximum value, b

Count Vector length, N

Definition at line 2521 of file DSPOp.cc.

5.10.3.155 void clDSPOp::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.

Definition at line 2555 of file DSPOp.cc.

5.10.3.156 void clDSPOp::Clip (float ∗, const float ∗, float, float, long) `[static]`

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

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

Parameters:

Dest Destination, y()

Src Source, x()

Min Minimum value, a

Max Maximum value, b

Count Vector length, N

Definition at line 2589 of file DSPOp.cc.

5.10.3.157 void clDSPOp::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.

Definition at line 2632 of file DSPOp.cc.

5.10.3.158 void clDSPOp::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

Definition at line 2675 of file DSPOp.cc.

Referenced by clDSPVector< TDSPVector_t >::ClipZero(), dsp_clipzero(), dsp_-clipzero_nip(), dsp_clipzerof(), and dsp_clipzerof_nip().

5.10.3.159 void clDSPOp::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.

Definition at line 2700 of file DSPOp.cc.

5.10.3.160 void clDSPOp::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

Definition at line 2725 of file DSPOp.cc.

5.10.3.161 void clDSPOp::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.

Definition at line 2746 of file DSPOp.cc.

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

Copy source vector to destination vector(s).

Parameters:

Dest Destination vector

Src Source vector

Count Vector length

Definition at line 2767 of file DSPOp.cc.

References likely.

Referenced by DCTo(), dsp_ccopy(), dsp_ccopyf(), dsp_copy(), dsp_copyf(), FIRAllocate(), FIRFilter(), FIRFilterF(), clRingBuffer< TRingBuffer_t >::Get(), clFilter::GetCoeffs(), IIRInitialize(), clFilter::Initialize(), Median(), clRingBuffer< TRingBuffer_t >::Put(), clFilter::Put(), ReBuffer(), ResampleAvg(), and clFilter::SetCoeffs().

5.10.3.163 void clDSPOp::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.

Definition at line 2819 of file DSPOp.cc.

References likely.

5.10.3.164 void clDSPOp::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.

Definition at line 2871 of file DSPOp.cc.

References likely, stpSCplx, and stSCplx.

5.10.3.165 void clDSPOp::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.

Definition at line 2925 of file DSPOp.cc.

References likely, stDCplx, and stpDCplx.

5.10.3.166 void clDSPOp::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.

Definition at line 2979 of file DSPOp.cc.

5.10.3.167 void clDSPOp::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.

Definition at line 2996 of file DSPOp.cc.

5.10.3.168 float clDSPOp::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

Definition at line 3013 of file DSPOp.cc.

Referenced by clDSPVector< TDSPVector_t >::Conv(), dsp_convolve(), dsp_-convolve2(), dsp_convolve2f(), and dsp_convolvef().

5.10.3.169 double clDSPOp::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.

Definition at line 3033 of file DSPOp.cc.

5.10.3.170 void clDSPOp::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

Definition at line 3053 of file DSPOp.cc.

5.10.3.171 void clDSPOp::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.

Definition at line 3081 of file DSPOp.cc.

5.10.3.172 float clDSPOp::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

Definition at line 3109 of file DSPOp.cc.

Referenced by clDSPVector$<$ TDSPVector_t $>$::Corr(), dsp_correlate(), dsp_-correlate2(), dsp_correlate2f(), and dsp_correlatef().

5.10.3.173 double clDSPOp::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.

Definition at line 3128 of file DSPOp.cc.

5.10.3.174 void clDSPOp::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

Definition at line 3147 of file DSPOp.cc.

5.10.3.175 void clDSPOp::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.

Definition at line 3175 of file DSPOp.cc.

5.10.3.176 float clDSPOp::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)

Definition at line 3203 of file DSPOp.cc.

Referenced by clDSPVector< TDSPVector_t >::AutoCorr(), dsp_autocorr(), dsp_autocorr2(), dsp_autocorr2f(), and dsp_autocorrf().

5.10.3.177 double clDSPOp::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.

Definition at line 3221 of file DSPOp.cc.

5.10.3.178 void clDSPOp::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

Definition at line 3239 of file DSPOp.cc.

5.10.3.179 void clDSPOp::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.

Definition at line 3266 of file DSPOp.cc.

5.10.3.180 float clDSPOp::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

Definition at line 3293 of file DSPOp.cc.

Referenced by clDSPVector< TDSPVector_t >::DotProduct(), dsp_dotproduct(), and dsp_dotproductf().

5.10.3.181 double clDSPOp::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.

Definition at line 3317 of file DSPOp.cc.

5.10.3.182 void clDSPOp::MinMax (float ∗, float ∗, const float ∗, long) `[static]`

Find minimum and maximum values of vector.

Parameters:
Min Minimum
Max Maximum
Src Source
Count Vector length

Definition at line 3341 of file DSPOp.cc.

Referenced by dsp_minmax(), dsp_minmaxf(), clDSPVector< TDSPVector_t >::MinMax(), PeakLevel(), Scale(), Scale01(), and WinDolphChebyshev().

5.10.3.183 void clDSPOp::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.

Definition at line 3394 of file DSPOp.cc.

5.10.3.184 float clDSPOp::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

Definition at line 3443 of file DSPOp.cc.

Referenced by dsp_mean(), dsp_meanf(), and clDSPVector< TDSPVector_-t >::Mean().

5.10.3.185 double clDSPOp::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.

Definition at line 3463 of file DSPOp.cc.

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

Calculate median of vector.

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

Parameters:

Src Source, x()

Count Vector length, N

Returns:

Median, m

Definition at line 3483 of file DSPOp.cc.

References Copy(), and Sort().

Referenced by dsp_median(), dsp_medianf(), and clDSPVector< TDSPVector_-t >::Median().

5.10.3.187 double clDSPOp::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.

Definition at line 3509 of file DSPOp.cc.

References Copy(), and Sort().

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

Negate vector elements.

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

Parameters:

Vect Source & destination

Count Vector length

Definition at line 3535 of file DSPOp.cc.

Referenced by dsp_negate(), dsp_negate_nip(), dsp_negatef(), dsp_negatef_nip(), and clDSPVector< TDSPVector_t >::Negate().

5.10.3.189 void clDSPOp::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.

Definition at line 3546 of file DSPOp.cc.

5.10.3.190 void clDSPOp::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

Definition at line 3557 of file DSPOp.cc.

5.10.3.191 void clDSPOp::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.

Definition at line 3568 of file DSPOp.cc.

5.10.3.192 void clDSPOp::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

Definition at line 3579 of file DSPOp.cc.

References StdDev().

Referenced by dsp_normalize(), dsp_normalize_nip(), dsp_normalizef(), dsp_-normalizef_nip(), and clDSPVector< TDSPVector_t >::Normalize().

5.10.3.193 void clDSPOp::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.

Definition at line 3593 of file DSPOp.cc.

References StdDev().

5.10.3.194 void clDSPOp::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

Definition at line 3607 of file DSPOp.cc.

References StdDev().

5.10.3.195 void clDSPOp::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.

Definition at line 3621 of file DSPOp.cc.

References StdDev().

5.10.3.196 float clDSPOp::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

Definition at line 3635 of file DSPOp.cc.

Referenced by dsp_product(), and dsp_productf().

5.10.3.197 double clDSPOp::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.

Definition at line 3648 of file DSPOp.cc.

5.10.3.198 void clDSPOp::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

Definition at line 3661 of file DSPOp.cc.

Referenced by dsp_reverse(), dsp_reverse_nip(), dsp_reversef(), dsp_reversef_nip(), clFlipBand::Put(), and clDSPVector< TDSPVector_t >::Reverse().

5.10.3.199 void clDSPOp::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.

Definition at line 3688 of file DSPOp.cc.

5.10.3.200 void clDSPOp::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.

Definition at line 3715 of file DSPOp.cc.

References CplxConj(), _sSCplx::I, _sSCplx::R, stpSCplx, and stSCplx.

5.10.3.201 void clDSPOp::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.

Definition at line 3753 of file DSPOp.cc.

References CplxConj(), _sDCplx::I, _sDCplx::R, stDCplx, and stpDCplx.

5.10.3.202 void clDSPOp::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

Definition at line 3791 of file DSPOp.cc.

5.10.3.203 void clDSPOp::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.

Definition at line 3808 of file DSPOp.cc.

5.10.3.204 void clDSPOp::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.

Definition at line 3825 of file DSPOp.cc.

References CplxConj(), and stpSCplx.

5.10.3.205 void clDSPOp::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.

Definition at line 3842 of file DSPOp.cc.

References CplxConj(), and stpDCplx.

5.10.3.206 void clDSPOp::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

Definition at line 3859 of file DSPOp.cc.

References MinMax().

Referenced by dsp_scale(), dsp_scale_nip(), dsp_scalef(), dsp_scalef_nip(), and clDSPVector< TDSPVector_t >::Scale().

5.10.3.207 void clDSPOp::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.

Definition at line 3905 of file DSPOp.cc.

References MinMax().

5.10.3.208 void clDSPOp::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

Definition at line 3947 of file DSPOp.cc.

References MinMax().

5.10.3.209 void clDSPOp::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.

Definition at line 3993 of file DSPOp.cc.

References MinMax().

5.10.3.210 void clDSPOp::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

Definition at line 4035 of file DSPOp.cc.

References MinMax().

Referenced by dsp_scale01(), dsp_scale01_nip(), dsp_scale01f(), dsp_scale01f_nip(), and clDSPVector< TDSPVector_t >::Scale01().

5.10.3.211 void clDSPOp::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.

Definition at line 4079 of file DSPOp.cc.

References MinMax().

5.10.3.212 void clDSPOp::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

Definition at line 4119 of file DSPOp.cc.

References MinMax().

5.10.3.213 void clDSPOp::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.

Definition at line 4163 of file DSPOp.cc.

References MinMax().

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

Sort vector elements (in-place).

Vector is sorted using quick-sort algorithm.

Parameters:

Vect Source & destination

Count Vector length

Definition at line 4203 of file DSPOp.cc.

References FloatCompare().

Referenced by dsp_sort(), dsp_sortf(), dsp_sortl(), Median(), and clDSPVector$<$ TDSPVector_t $>$::Sort().

5.10.3.215 void clDSPOp::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.

Definition at line 4213 of file DSPOp.cc.

References DoubleCompare().

5.10.3.216 void clDSPOp::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.

Definition at line 4223 of file DSPOp.cc.

References LongCompare().

5.10.3.217 void clDSPOp::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

Definition at line 4233 of file DSPOp.cc.

Referenced by dsp_stddev(), dsp_stddevf(), Normalize(), and clDSPVector< TDSPVector_t >::StdDev().

5.10.3.218 void clDSPOp::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.

Definition at line 4267 of file DSPOp.cc.

5.10.3.219 float clDSPOp::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

Definition at line 4301 of file DSPOp.cc.

Referenced by dsp_sum(), dsp_sumf(), and clDSPVector< TDSPVector_t >::Sum().

5.10.3.220 double clDSPOp::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.

Definition at line 4320 of file DSPOp.cc.

5.10.3.221 void clDSPOp::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

Definition at line 4339 of file DSPOp.cc.

Referenced by dsp_square(), dsp_square_nip(), dsp_squaref(), dsp_squaref_nip(), and clDSPVector< TDSPVector_t >::Square().

5.10.3.222 void clDSPOp::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.

Definition at line 4354 of file DSPOp.cc.

5.10.3.223 void clDSPOp::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

Definition at line 4369 of file DSPOp.cc.

5.10.3.224 void clDSPOp::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.

Definition at line 4384 of file DSPOp.cc.

5.10.3.225 void clDSPOp::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

Definition at line 4399 of file DSPOp.cc.

Referenced by clDSPVector< TDSPVector_t >::Convert(), dsp_convertd64c(), dsp_convertd64f(), dsp_convertd64i(), dsp_convertd64s(), dsp_convertf32(), dsp_convertf32c(), dsp_convertf32i(), dsp_convertf32s(), dsp_converts16(), dsp_converts16f(), dsp_converts32(), dsp_converts32f(), dsp_convertu8(), dsp_-convertu8f(), and clFilter::Initialize().

5.10.3.226 void clDSPOp::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

Definition at line 4412 of file DSPOp.cc.

5.10.3.227 void clDSPOp::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

Definition at line 4441 of file DSPOp.cc.

5.10.3.228 void clDSPOp::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.

Definition at line 4472 of file DSPOp.cc.

5.10.3.229 void clDSPOp::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.

Definition at line 4487 of file DSPOp.cc.

5.10.3.230 void clDSPOp::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.

Definition at line 4501 of file DSPOp.cc.

5.10.3.231 void clDSPOp::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.

Definition at line 4520 of file DSPOp.cc.

5.10.3.232 void clDSPOp::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.

Definition at line 4541 of file DSPOp.cc.

5.10.3.233 void clDSPOp::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.

Definition at line 4556 of file DSPOp.cc.

References Round().

5.10.3.234 void clDSPOp::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.

Definition at line 4568 of file DSPOp.cc.

References Round().

5.10.3.235 void clDSPOp::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.

Definition at line 4581 of file DSPOp.cc.

References Round().

5.10.3.236 void clDSPOp::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.

Definition at line 4601 of file DSPOp.cc.

References Round().

5.10.3.237 void clDSPOp::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.

Definition at line 4616 of file DSPOp.cc.

References Round().

5.10.3.238 void clDSPOp::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.

Definition at line 4635 of file DSPOp.cc.

References Round().

5.10.3.239 void clDSPOp::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

Definition at line 4654 of file DSPOp.cc.

References Cart2Polar().

Referenced by clDSPVector< TDSPVector_t >::CartToPolar(), dsp_cart2polar(), dsp_cart2polar2(), dsp_cart2polar2f(), dsp_cart2polar3(), dsp_cart2polar3f(), dsp_-cart2polar4(), dsp_cart2polar4f(), and dsp_cart2polarf().

5.10.3.240 void clDSPOp::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.

Definition at line 4671 of file DSPOp.cc.

References Cart2Polar().

5.10.3.241 void clDSPOp::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

Definition at line 4688 of file DSPOp.cc.

References Cart2Polar(), and stpSCplx.

5.10.3.242 void clDSPOp::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.

Definition at line 4705 of file DSPOp.cc.

References Cart2Polar(), and stpDCplx.

5.10.3.243 void clDSPOp::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

Definition at line 4722 of file DSPOp.cc.

References Cart2Polar(), stpSCplx, and stpSPolar.

5.10.3.244 void clDSPOp::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.

Definition at line 4734 of file DSPOp.cc.

References Cart2Polar(), stpDCplx, and stpDPolar.

5.10.3.245 void clDSPOp::CartToPolar (utpSCoord, long) `[static]`

Convert cartesian to polar vectors (in-place).

Definition at line 4746 of file DSPOp.cc.

References Cart2Polar(), and utpSCoord.

5.10.3.246 void clDSPOp::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.

Definition at line 4757 of file DSPOp.cc.

References Cart2Polar(), and utpDCoord.

5.10.3.247 void clDSPOp::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

Definition at line 4768 of file DSPOp.cc.

References Polar2Cart().

Referenced by dsp_polar2cart(), dsp_polar2cart2(), dsp_polar2cart2f(), dsp_polar2cart3(), dsp_polar2cart3f(), dsp_polar2cart4(), dsp_polar2cart4f(), dsp_polar2cartf(), and clDSPVector< TDSPVector_t >::PolarToCart().

5.10.3.248 void clDSPOp::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.

Definition at line 4785 of file DSPOp.cc.

References Polar2Cart().

5.10.3.249 void clDSPOp::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

Definition at line 4802 of file DSPOp.cc.

References Polar2Cart(), and stpSCplx.

5.10.3.250 void clDSPOp::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.

Definition at line 4818 of file DSPOp.cc.

References Polar2Cart(), and stpDCplx.

5.10.3.251 void clDSPOp::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

Definition at line 4834 of file DSPOp.cc.

References Polar2Cart(), stpSCplx, and stpSPolar.

5.10.3.252 void clDSPOp::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.

Definition at line 4846 of file DSPOp.cc.

References Polar2Cart(), stpDCplx, and stpDPolar.

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

Convert cartesian to polar vectors (in-place).

Definition at line 4858 of file DSPOp.cc.

References Polar2Cart(), and utpSCoord.

5.10.3.254 void clDSPOp::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.

Definition at line 4869 of file DSPOp.cc.

References Polar2Cart(), and utpDCoord.

5.10.3.255 float clDSPOp::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

Definition at line 4880 of file DSPOp.cc.

Referenced by clDSPVector< TDSPVector_t >::CrossCorr(), dsp_crosscorr(), and dsp_crosscorrf().

5.10.3.256 double clDSPOp::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.

Definition at line 4944 of file DSPOp.cc.

5.10.3.257 float clDSPOp::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

Definition at line 5004 of file DSPOp.cc.

Referenced by clDSPVector< TDSPVector_t >::CrossCorr(), dsp_crosscorr2(), dsp_-crosscorr2f(), dsp_crosscorr3(), and dsp_crosscorr3f().

5.10.3.258 double clDSPOp::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.

Definition at line 5054 of file DSPOp.cc.

5.10.3.259 void clDSPOp::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

Definition at line 5100 of file DSPOp.cc.

5.10.3.260 void clDSPOp::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.

Definition at line 5141 of file DSPOp.cc.

5.10.3.261 float clDSPOp::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

Definition at line 5182 of file DSPOp.cc.

Referenced by dsp_energy(), dsp_energyf(), and clDSPVector< TDSPVector_t >::Energy().

5.10.3.262 double clDSPOp::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.

Definition at line 5199 of file DSPOp.cc.

5.10.3.263 void clDSPOp::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

Definition at line 5216 of file DSPOp.cc.

References stpSCplx.

Referenced by dsp_magnitude(), dsp_magnitudef(), and clDSPVector< TDSPVector_t >::Magnitude().

5.10.3.264 void clDSPOp::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.

Definition at line 5238 of file DSPOp.cc.

References stpDCplx.

5.10.3.265 void clDSPOp::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

Definition at line 5260 of file DSPOp.cc.

References stpSCplx.

Referenced by dsp_power(), dsp_powerf(), and clDSPVector< TDSPVector_-t >::Power().

5.10.3.266 void clDSPOp::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.

Definition at line 5278 of file DSPOp.cc.

References stpDCplx.

5.10.3.267 void clDSPOp::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

Definition at line 5296 of file DSPOp.cc.

References stpSCplx.

Referenced by dsp_phase(), dsp_phasef(), and clDSPVector< TDSPVector_-t >::Phase().

5.10.3.268 void clDSPOp::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.

Definition at line 5312 of file DSPOp.cc.

References stpDCplx.

5.10.3.269 void clDSPOp::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

Definition at line 5329 of file DSPOp.cc.

References _sSCplx::R, and stpSCplx.

Referenced by dsp_powerphase(), dsp_powerphasef(), and clDSPVector< TDSPVector_t >::PowerPhase().

5.10.3.270 void clDSPOp::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.

Definition at line 5352 of file DSPOp.cc.

References _sDCplx::R, and stpDCplx.

5.10.3.271 void clDSPOp::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)

Definition at line 5375 of file DSPOp.cc.

Referenced by clDSPVector< TDSPVector_t >::Decimate(), dsp_decimate(), dsp_decimatef(), clIIRDecimator::Get(), clFIRDecimator::Get(), and clFFTDecimator::Get().

5.10.3.272 void clDSPOp::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.

Definition at line 5396 of file DSPOp.cc.

5.10.3.273 void clDSPOp::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)

Definition at line 5417 of file DSPOp.cc.

Referenced by clDSPVector< TDSPVector_t >::DecimateAvg(), dsp_decimateavg(), and dsp_decimateavgf().

5.10.3.274 void clDSPOp::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.

Definition at line 5438 of file DSPOp.cc.

5.10.3.275 void clDSPOp::Interpolate (float ∗, const float ∗, long, long) [static]

Iterpolate vector without average (nulling).

Parameters:

Dest Destination

Src Source

Factor Interpolation factor

Count Vector length (source)

Definition at line 5459 of file DSPOp.cc.

Referenced by dsp_interpolate(), dsp_interpolatef(), clDSPVector$<$ TDSPVector_t $>$::Interpolate(), clIIRInterpolator::Put(), clFIRInterpolator::Put(), and clFFTInterpolator::Put().

5.10.3.276 void clDSPOp::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.

Definition at line 5483 of file DSPOp.cc.

5.10.3.277 void clDSPOp::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)

Definition at line 5507 of file DSPOp.cc.

Referenced by dsp_interpolateavg(), dsp_interpolateavgf(), and clDSPVector$<$ TDSPVector_t $>$::InterpolateAvg().

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

Definition at line 5548 of file DSPOp.cc.

5.10.3.279 void clDSPOp::Resample (float $*$, long, const float $*$, long) [static]

Resample vector to different length.

Parameters:

Dest Destination

DestCount Destionation vector length

Src Source

SrcCount Source vector length

Definition at line 5589 of file DSPOp.cc.

References unlikely.

Referenced by dsp_resample(), dsp_resamplef(), and clDSPVector< TDSPVector_t >::Resample().

5.10.3.280 void clDSPOp::Resample (double $*$, long, const double $*$, long) [static]

Definition at line 5607 of file DSPOp.cc.

References unlikely.

5.10.3.281 void clDSPOp::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

Definition at line 5625 of file DSPOp.cc.

References Copy(), and unlikely.

Referenced by dsp_resampleavg(), dsp_resampleavgf(), and clDSPVector< TDSPVector_t >::ResampleAvg().

5.10.3.282 void clDSPOp::ResampleAvg (double $*$, long, const double $*$, long) [static]

Definition at line 5678 of file DSPOp.cc.

References Copy(), and unlikely.

5.10.3.283 float clDSPOp::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

Definition at line 5731 of file DSPOp.cc.

Referenced by dsp_rms(), dsp_rmsf(), and clDSPVector< TDSPVector_t >::RMS().

5.10.3.284 double clDSPOp::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.

Definition at line 5748 of file DSPOp.cc.

5.10.3.285 float clDSPOp::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

Definition at line 5765 of file DSPOp.cc.

Referenced by dsp_variance(), dsp_variancef(), and clDSPVector< TDSPVector_-t >::Variance().

5.10.3.286 double clDSPOp::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.

Definition at line 5794 of file DSPOp.cc.

5.10.3.287 float clDSPOp::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)

Definition at line 5823 of file DSPOp.cc.

References MinMax().

Referenced by dsp_peaklevel(), dsp_peaklevelf(), and clDSPVector< TDSPVector_t >::PeakLevel().

5.10.3.288 double clDSPOp::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.

Definition at line 5841 of file DSPOp.cc.

References MinMax().

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

Calculate Bartlett (triangle) window coefficients.

Parameters:
Dest Destination

Count Vector length

Definition at line 5859 of file DSPOp.cc.

Referenced by clDSPVector< TDSPVector_t >::WinBartlett().

5.10.3.290 void clDSPOp::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.

Definition at line 5880 of file DSPOp.cc.

5.10.3.291 void clDSPOp::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

Definition at line 5901 of file DSPOp.cc.

References fPI.

Referenced by clDSPVector< TDSPVector_t >::WinBlackman().

5.10.3.292 void clDSPOp::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.

Definition at line 5919 of file DSPOp.cc.

References dPI.

5.10.3.293 void clDSPOp::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)

Definition at line 5937 of file DSPOp.cc.

References fPI.

5.10.3.294 void clDSPOp::WinBlackman (double *, long, double)

Definition at line 5963 of file DSPOp.cc.

References dPI.

5.10.3.295 void clDSPOp::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

Definition at line 5989 of file DSPOp.cc.

References fPI.

Referenced by clDSPVector< TDSPVector_t >::WinBlackmanHarris().

5.10.3.296 void clDSPOp::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.

Definition at line 6004 of file DSPOp.cc.

References dPI.

5.10.3.297 void clDSPOp::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

Definition at line 6019 of file DSPOp.cc.

References fPI, and Round().

Referenced by clDSPVector< TDSPVector_t >::WinCosTapered().

5.10.3.298 void clDSPOp::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.

Definition at line 6040 of file DSPOp.cc.

References dPI, and Round().

5.10.3.299 void clDSPOp::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)

Definition at line 6061 of file DSPOp.cc.

References fPI, and Round().

5.10.3.300 void clDSPOp::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.

Definition at line 6080 of file DSPOp.cc.

References dPI, and Round().

5.10.3.301 void clDSPOp::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)

Definition at line 6099 of file DSPOp.cc.

References fPI, and Round().

5.10.3.302 void clDSPOp::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.

Definition at line 6120 of file DSPOp.cc.

References dPI, and Round().

5.10.3.303 void clDSPOp::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

Definition at line 6141 of file DSPOp.cc.

References fPI.

Referenced by clDSPVector< TDSPVector_t >::WinExactBlackman().

5.10.3.304 void clDSPOp::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.

Definition at line 6160 of file DSPOp.cc.

References dPI.

5.10.3.305 void clDSPOp::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

Definition at line 6179 of file DSPOp.cc.

Referenced by clDSPVector< TDSPVector_t >::WinExp().

5.10.3.306 void clDSPOp::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.

Definition at line 6193 of file DSPOp.cc.

5.10.3.307 void clDSPOp::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

Definition at line 6207 of file DSPOp.cc.

References fPI.

Referenced by clDSPVector< TDSPVector_t >::WinFlatTop().

5.10.3.308 void clDSPOp::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.

Definition at line 6222 of file DSPOp.cc.

References dPI.

5.10.3.309 void clDSPOp::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

Definition at line 6237 of file DSPOp.cc.

References fPI.

Referenced by clDSPVector< TDSPVector_t >::WinGenericCos().

5.10.3.310 void clDSPOp::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.

Definition at line 6259 of file DSPOp.cc.

References dPI.

5.10.3.311 void clDSPOp::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

Definition at line 6281 of file DSPOp.cc.

References fPI.

Referenced by clDSPVector< TDSPVector_t >::WinHamming().

5.10.3.312 void clDSPOp::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.

Definition at line 6298 of file DSPOp.cc.

References dPI.

5.10.3.313 void clDSPOp::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

Definition at line 6315 of file DSPOp.cc.

References fPI.

Referenced by clDSPVector< TDSPVector_t >::WinHanning().

5.10.3.314 void clDSPOp::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.

Definition at line 6332 of file DSPOp.cc.

References dPI.

5.10.3.315 void clDSPOp::WinKaiser (float $*$, float, long)

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

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

Parameters:

Dest Destination, y()

Beta Beta, β

Count Vector length, N

Definition at line 6349 of file DSPOp.cc.

References ModZeroBessel().

Referenced by clFilter::Initialize(), and clDSPVector< TDSPVector_t >::WinKaiser().

5.10.3.316 void clDSPOp::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.

Definition at line 6370 of file DSPOp.cc.

References ModZeroBessel().

5.10.3.317 void clDSPOp::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

Definition at line 6391 of file DSPOp.cc.

References fPI, and ModZeroBessel().

Referenced by clFilter::Initialize(), and clDSPVector< TDSPVector_t >::WinKaiser-Bessel().

5.10.3.318 void clDSPOp::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.

Definition at line 6409 of file DSPOp.cc.

References dPI, and ModZeroBessel().

5.10.3.319 void clDSPOp::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

Definition at line 6427 of file DSPOp.cc.

References fPI.

Referenced by clDSPVector< TDSPVector_t >::WinTukey().

5.10.3.320 void clDSPOp::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.

Definition at line 6441 of file DSPOp.cc.

References dPI.

5.10.3.321 void clDSPOp::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

Definition at line 6455 of file DSPOp.cc.

References ChebyshevPolynom(), fPI, MinMax(), and Mul().

Referenced by clFilter::Initialize(), and clDSPVector< TDSPVector_t >::WinDolph-Chebyshev().

5.10.3.322 void clDSPOp::WinDolphChebyshev (double ∗, double, long)

Definition at line 6498 of file DSPOp.cc.

References ChebyshevPolynom(), dPI, MinMax(), and Mul().

5.10.3.323 void clDSPOp::Mix (float ∗, const float ∗, long) `[static]`

Mix two channels interleaved in vector.

Parameters:

Dest Destination

Src Source

Count Vector length (destination)

Definition at line 6541 of file DSPOp.cc.

Referenced by dsp_mix(), dsp_mix2(), dsp_mix2f(), dsp_mixf(), dsp_mixn(), dsp_-mixnf(), and clDSPVector< TDSPVector_t >::Mix().

5.10.3.324 void clDSPOp::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.

Definition at line 6554 of file DSPOp.cc.

5.10.3.325 void clDSPOp::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

Definition at line 6567 of file DSPOp.cc.

5.10.3.326 void clDSPOp::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.

Definition at line 6579 of file DSPOp.cc.

5.10.3.327 void clDSPOp::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

Definition at line 6591 of file DSPOp.cc.

5.10.3.328 void clDSPOp::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.

Definition at line 6614 of file DSPOp.cc.

5.10.3.329 void clDSPOp::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

Definition at line 6637 of file DSPOp.cc.

5.10.3.330 void clDSPOp::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.

Definition at line 6652 of file DSPOp.cc.

5.10.3.331 void clDSPOp::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.

Definition at line 6667 of file DSPOp.cc.

5.10.3.332 void clDSPOp::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.

Definition at line 6683 of file DSPOp.cc.

5.10.3.333 void clDSPOp::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)

Definition at line 6699 of file DSPOp.cc.

Referenced by dsp_extract(), dsp_extractf(), and clDSPVector< TDSPVector_t >::Extract().

5.10.3.334 void clDSPOp::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.

Definition at line 6713 of file DSPOp.cc.

5.10.3.335 void clDSPOp::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)

Definition at line 6727 of file DSPOp.cc.

Referenced by dsp_pack(), dsp_packf(), and clDSPVector< TDSPVector_t >::Pack().

5.10.3.336 void clDSPOp::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.

Definition at line 6739 of file DSPOp.cc.

5.10.3.337 long clDSPOp::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

Definition at line 6751 of file DSPOp.cc.

References Copy(), lPrevDestCount, and lPrevSrcCount.

5.10.3.338 long clDSPOp::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.

Definition at line 6788 of file DSPOp.cc.

References Copy(), lPrevDestCount, and lPrevSrcCount.

5.10.3.339 double clDSPOp::DegToRad (double *dSource*) `[inline]`

Convert degrees to radians.

$$y = \frac{\pi}{180} x$$

Parameters:

Source Degrees, x

Returns:

Radians, y

Definition at line 1925 of file DSPOp.hh.

References dPI.

5.10.3.340 float clDSPOp::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.

Definition at line 1928 of file DSPOp.hh.

References fPI.

5.10.3.341 double clDSPOp::RadToDeg (double *dSource*) `[inline]`

Convert radians to degrees.

$$y = \frac{180}{\pi} x$$

Parameters:

Source Radians, x

Degrees,y

Definition at line 1938 of file DSPOp.hh.

References dPI.

5.10.3.342 float clDSPOp::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.

Definition at line 1941 of file DSPOp.hh.

References fPI.

5.10.3.343 void clDSPOp::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

Definition at line 7894 of file DSPOp.cc.

References _sSCplx::I, _sSCplx::R, and stpSCplx.

Referenced by dsp_fftw_convertcd2d(), dsp_fftw_convertcd2f(), dsp_fftw_convertcf2d(), dsp_fftw_convertcf2f(), dsp_fftw_convertd2cd(), dsp_fftw_convertd2cf(), dsp_fftw_convertf2cd(), dsp_fftw_convertf2cf(), and clDSPVector< TDSPVector_t >::FFTWConvert().

5.10.3.344 void clDSPOp::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.

Definition at line 7912 of file DSPOp.cc.

References _sDCplx::I, _sDCplx::R, and stpDCplx.

5.10.3.345 void clDSPOp::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.

Definition at line 7930 of file DSPOp.cc.

References _sSCplx::I, _sSCplx::R, and stpSCplx.

5.10.3.346 void clDSPOp::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.

Definition at line 7948 of file DSPOp.cc.

References _sDCplx::I, _sDCplx::R, and stpDCplx.

5.10.3.347 void clDSPOp::FFTWConvert (float $*$, const *stpSCplx*, long) [static]

Convert our Cplx to Real-FFTW complex input.

Definition at line 7966 of file DSPOp.cc.

References _sSCplx::R, and stpSCplx.

5.10.3.348 void clDSPOp::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.

Definition at line 7982 of file DSPOp.cc.

References _sDCplx::R, and stpDCplx.

5.10.3.349 void clDSPOp::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.

Definition at line 7998 of file DSPOp.cc.

References _sSCplx::R, and stpSCplx.

5.10.3.350 void clDSPOp::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.

Definition at line 8014 of file DSPOp.cc.

References _sDCplx::R, and stpDCplx.

5.10.3.351 void clDSPOp::FIRAllocate (const float $*$, long)

Allocate FIR filter.

Parameters:

Coeff Coefficients

Count Vector length

Definition at line 6828 of file DSPOp.cc.

References Copy(), FIRBuf, FIRCoeff, iFIRDlyIdx, lFIRLength, clAlloc::Size(), and Zero().

Referenced by clDSPVector< TDSPVector_t >::FIRAllocate(), and clFIRMultiRate::Initialize().

5.10.3.352 void clDSPOp::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.

Definition at line 6847 of file DSPOp.cc.

References Copy(), FIRBuf, FIRCoeff, iFIRDlyIdx, lFIRLength, clAlloc::Size(), and Zero().

5.10.3.353 void clDSPOp::FIRFilter (float *, long)

Filter using FIR filter.

Parameters:

Vect Source & destination

Count Vector length

Definition at line 6866 of file DSPOp.cc.

References Copy(), FIRBuf, FIRCoeff, FIRWork, iFIRDlyIdx, lFIRLength, and clAlloc::Size().

Referenced by clDSPVector< TDSPVector_t >::FIRFilter(), clFIRDecimator::Get(), and clFIRInterpolator::Put().

5.10.3.354 void clDSPOp::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.

Definition at line 6926 of file DSPOp.cc.

References Copy(), FIRBuf, FIRCoeff, FIRWork, iFIRDlyIdx, lFIRLength, and clAlloc::Size().

5.10.3.355 void clDSPOp::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

Definition at line 6981 of file DSPOp.cc.

References Copy(), FIRBuf, FIRCoeff, FIRWork, iFIRDlyIdx, lFIRLength, and clAlloc::Size().

5.10.3.356 void clDSPOp::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.

Definition at line 7041 of file DSPOp.cc.

References Copy(), FIRBuf, FIRCoeff, FIRWork, iFIRDlyIdx, lFIRLength, and clAlloc::Size().

5.10.3.357 void clDSPOp::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

Definition at line 7096 of file DSPOp.cc.

References Copy(), FIRCoeff, and lFIRLength.

5.10.3.358 void clDSPOp::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.

Definition at line 7126 of file DSPOp.cc.

References Copy(), FIRCoeff, and lFIRLength.

5.10.3.359 void clDSPOp::FIRFree ()

Free FIR filter.

Definition at line 7156 of file DSPOp.cc.

References FIRBuf, FIRCoeff, and clAlloc::Free().

Referenced by clDSPVector< TDSPVector_t >::FIRFree(), and clFIRMultiRate::Uninitialize().

5.10.3.360 void clDSPOp::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

Definition at line 7163 of file DSPOp.cc.

References Copy(), and fpIIR_C.

Referenced by clIIRCascade::Initialize().

5.10.3.361 void clDSPOp::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.

Definition at line 7169 of file DSPOp.cc.

References Copy(), and dpIIR_C.

5.10.3.362 void clDSPOp::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

Definition at line 7175 of file DSPOp.cc.

References fpIIR_C, fpIIR_X, and fpIIR_Y.

Referenced by clIIRCascade::Process().

5.10.3.363 void clDSPOp::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.

Definition at line 7217 of file DSPOp.cc.

References dpIIR_C, dpIIR_X, and dpIIR_Y.

5.10.3.364 void clDSPOp::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

Definition at line 7255 of file DSPOp.cc.

References fpIIR_C, fpIIR_X, and fpIIR_Y.

5.10.3.365 void clDSPOp::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.

Definition at line 7299 of file DSPOp.cc.

References dpIIR_C, dpIIR_X, and dpIIR_Y.

5.10.3.366 void clDSPOp::IIRClear ()

Clear IIR filter feedback chain.

Definition at line 7337 of file DSPOp.cc.

References dpIIR_X, dpIIR_Y, fpIIR_X, fpIIR_Y, and Zero().

Referenced by clIIRCascade::Clear().

5.10.3.367 void clDSPOp::FFTInitialize (long, bool)

Initialize FFT.

Parameters:

Size Transform size

Real Real transform?

Definition at line 7346 of file DSPOp.cc.

References bFFTInitialized, bRealTransform, clTransformS::cdft(), DBitRevWork, DCosSinTable, dFFTScale, dpCosSinTable, fFFTScale, FFTBuf, fpCosSinTable, l-FFTLength, lpDBitRevWork, lpSBitRevWork, clTransformS::rdft(), SBitRevWork, SCosSinTable, clAlloc::Size(), Tfrm, vpDTfrm, and vpSTfrm.

Referenced by clDSPVector< TDSPVector_t >::FFTInitialize(), clHankel::Initialize(), clFlipBand::Initialize(), and clFilter::Initialize().

5.10.3.368 void clDSPOp::FFTUninitialize ()

Uninitialize FFT.

Definition at line 7469 of file DSPOp.cc.

References bFFTInitialized, bRealTransform, DBitRevWork, DCosSinTable, FFTBuf, clAlloc::Free(), SBitRevWork, SCosSinTable, vpDTfrm, and vpSTfrm.

Referenced by clDSPVector< TDSPVector_t >::FFTUninitialize(), clFlipBand::Uninitialize(), clFilter::Uninitialize(), and ∼clDSPOp().

5.10.3.369 void clDSPOp::FFTi (stpSCplx, float ∗)

FFT half in-place, source vector is modified.

Note:

Output is FFT size / 2 + 1

Parameters:

Dest Destination

Src Source

Definition at line 7505 of file DSPOp.cc.

References fFFTScale, fpCosSinTable, lFFTLength, lpSBitRevWork, Mul(), clTransformS::rdft(), SBitRevWork, stpSCplx, Tfrm, and vpSTfrm.

Referenced by clFlipBand::Put(), clFilter::Put(), clFilter::ReadyFilterD(), and clFilter::ReadyFilterS().

5.10.3.370 void clDSPOp::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.

Definition at line 7533 of file DSPOp.cc.

References DBitRevWork, dFFTScale, dpCosSinTable, lFFTLength, lpDBitRevWork, Mul(), clTransformS::rdft(), stpDCplx, Tfrm, and vpDTfrm.

5.10.3.371 void clDSPOp::FFTo (stpSCplx, const float ∗)

FFT out-of-place.

Note:

Output is FFT size / 2 + 1 for real input

Parameters:

Dest Destination

Src Source

Definition at line 7561 of file DSPOp.cc.

References fFFTScale, FFTBuf, fpCosSinTable, lFFTLength, lpSBitRevWork, Mul(), clTransformS::rdft(), SBitRevWork, stpSCplx, Tfrm, and vpSTfrm.

Referenced by clDSPVector< TDSPVector_t >::FFT().

5.10.3.372 void clDSPOp::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.

Definition at line 7594 of file DSPOp.cc.

References DBitRevWork, dFFTScale, dpCosSinTable, FFTBuf, lFFTLength, lpDBitRevWork, Mul(), clTransformS::rdft(), stpDCplx, Tfrm, and vpDTfrm.

5.10.3.373 void clDSPOp::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.

Definition at line 7627 of file DSPOp.cc.

References clTransformS::cdft(), fFFTScale, FFTBuf, fpCosSinTable, lFFTLength, lpSBitRevWork, Mul(), SBitRevWork, stpSCplx, Tfrm, and vpSTfrm.

5.10.3.374 void clDSPOp::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.

Definition at line 7656 of file DSPOp.cc.

References clTransformS::cdft(), DBitRevWork, dFFTScale, dpCosSinTable, FFTBuf, lFFTLength, lpDBitRevWork, Mul(), stpDCplx, Tfrm, and vpDTfrm.

5.10.3.375 void clDSPOp::IFFTo (float ∗, const *stpSCplx*)

IFFT out-of-place.

Note:

Input is FFT size / 2 + 1 for real output

Parameters:

Dest Destination

Src Source

Definition at line 7685 of file DSPOp.cc.

References fpCosSinTable, lFFTLength, lpSBitRevWork, clTransformS::rdft(), SBitRevWork, stpSCplx, Tfrm, and vpSTfrm.

Referenced by clDSPVector< TDSPVector_t >::IFFT(), clHankel::Process0(), clHankel::Process1(), clFlipBand::Put(), clFilter::Put(), clFilter::ReadyFilterD(), and clFilter::ReadyFilterS().

5.10.3.376 void clDSPOp::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.

Definition at line 7709 of file DSPOp.cc.

References DBitRevWork, dpCosSinTable, lFFTLength, lpDBitRevWork, clTransformS::rdft(), stpDCplx, Tfrm, and vpDTfrm.

5.10.3.377 void clDSPOp::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.

Definition at line 7733 of file DSPOp.cc.

References clTransformS::cdft(), FFTBuf, fpCosSinTable, lFFTLength, lpSBitRevWork, SBitRevWork, stpSCplx, Tfrm, and vpSTfrm.

5.10.3.378 void clDSPOp::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.

Definition at line 7763 of file DSPOp.cc.

References clTransformS::cdft(), DBitRevWork, dpCosSinTable, FFTBuf, lFFTLength, lpDBitRevWork, stpDCplx, Tfrm, and vpDTfrm.

5.10.3.379 void clDSPOp::DCTInitialize (long)

DCT initialize.

Note:

Internal state is shared with FFT

Parameters:

Size Transform size

Definition at line 7793 of file DSPOp.cc.

References bFFTInitialized, DBitRevWork, DCosSinTable, clTransformS::ddct(), dFFTScale, dpCosSinTable, fFFTScale, FFTBuf, fpCosSinTable, lFFTLength, lpDBitRevWork, lpSBitRevWork, SBitRevWork, SCosSinTable, clAlloc::Size(), and Tfrm.

Referenced by clDSPVector< TDSPVector_t >::DCTInitialize().

5.10.3.380 void clDSPOp::DCTUninitialize ()

Uninitialize DCT.

Definition at line 7825 of file DSPOp.cc.

References bFFTInitialized, DBitRevWork, DCosSinTable, FFTBuf, clAlloc::Free(), SBitRevWork, and SCosSinTable.

Referenced by clDSPVector< TDSPVector_t >::DCTUninitialize().

5.10.3.381 void clDSPOp::DCTi (float ∗)

DCT in-place.

Parameters:

Vect Source & destination vector

Definition at line 7836 of file DSPOp.cc.

References clTransformS::ddct(), fpCosSinTable, lFFTLength, lpSBitRevWork, and Tfrm.

Referenced by clDSPVector< TDSPVector_t >::DCT().

5.10.3.382 void clDSPOp::DCTi (double ∗)

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

Definition at line 7842 of file DSPOp.cc.

References clTransformS::ddct(), dpCosSinTable, lFFTLength, lpDBitRevWork, and Tfrm.

5.10.3.383 void clDSPOp::DCTo (float ∗, const float ∗)

DCT out-of-place.

Parameters:

Dest Destination vector

Src Source vector

Definition at line 7848 of file DSPOp.cc.

References Copy(), clTransformS::ddct(), fpCosSinTable, lFFTLength, lpSBitRevWork, and Tfrm.

Referenced by clDSPVector< TDSPVector_t >::DCT().

5.10.3.384 void clDSPOp::DCTo (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.

Definition at line 7855 of file DSPOp.cc.

References Copy(), clTransformS::ddct(), dpCosSinTable, lFFTLength, lpDBitRevWork, and Tfrm.

5.10.3.385 void clDSPOp::IDCTi (float *)

Inverse DCT in-place.

Parameters:

Vect Source & destination vector

Definition at line 7862 of file DSPOp.cc.

References clTransformS::ddct(), fFFTScale, fpCosSinTable, lFFTLength, lpSBitRevWork, Mul(), and Tfrm.

Referenced by clDSPVector< TDSPVector_t >::IDCT().

5.10.3.386 void clDSPOp::IDCTi (double *)

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

Definition at line 7870 of file DSPOp.cc.

References clTransformS::ddct(), dFFTScale, dpCosSinTable, lFFTLength, lpDBitRevWork, Mul(), and Tfrm.

5.10.3.387 void clDSPOp::IDCTo (float *, const float *)

Inverse DCT out-of-place.

Parameters:

Dest Destination vector

Src Source vector

Definition at line 7878 of file DSPOp.cc.

References clTransformS::ddct(), fFFTScale, fpCosSinTable, lFFTLength, lpSBitRevWork, Mul(), and Tfrm.

Referenced by clDSPVector< TDSPVector_t >::IDCT().

5.10.3.388 void clDSPOp::IDCTo (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.

Definition at line 7886 of file DSPOp.cc.

References clTransformS::ddct(), dFFTScale, dpCosSinTable, lFFTLength, lpDBitRevWork, Mul(), and Tfrm.

5.10.4 Member Data Documentation

5.10.4.1 long clDSPOp::lPrevSrcCount [private]

Definition at line 151 of file DSPOp.hh.

Referenced by clDSPOp(), and ReBuffer().

5.10.4.2 long clDSPOp::lPrevDestCount [private]

Definition at line 152 of file DSPOp.hh.

Referenced by clDSPOp(), and ReBuffer().

5.10.4.3 float clDSPOp::fPI [private]

Definition at line 153 of file DSPOp.hh.

Referenced by clDSPOp(), DegToRad(), RadToDeg(), WinBlackman(), WinBlackmanHarris(), WinCosTapered(), WinCosTaperedA(), WinDolphChebyshev(), WinExactBlackman(), WinFlatTop(), WinGenericCos(), WinHamming(), WinHanning(), WinKaiserBessel(), and WinTukey().

5.10.4.4 double clDSPOp::dPI [private]

Definition at line 154 of file DSPOp.hh.

Referenced by clDSPOp(), DegToRad(), RadToDeg(), WinBlackman(), WinBlackmanHarris(), WinCosTapered(), WinCosTaperedA(), WinDolphChebyshev(), WinExactBlackman(), WinFlatTop(), WinGenericCos(), WinHamming(), WinHanning(), WinKaiserBessel(), and WinTukey().

5.10.4.5 int clDSPOp::iFIRDlyIdx [private]

Definition at line 157 of file DSPOp.hh.

Referenced by FIRAllocate(), and FIRFilter().

5.10.4.6 long clDSPOp::lFIRLength [private]

Definition at line 158 of file DSPOp.hh.

Referenced by FIRAllocate(), FIRFilter(), and FIRFilterF().

5.10.4.7 clDSPAlloc clDSPOp::FIRCoeff [private]

Definition at line 159 of file DSPOp.hh.

Referenced by FIRAllocate(), FIRFilter(), FIRFilterF(), and FIRFree().

5.10.4.8 clDSPAlloc clDSPOp::FIRBuf [private]

Definition at line 160 of file DSPOp.hh.

Referenced by FIRAllocate(), FIRFilter(), and FIRFree().

5.10.4.9 clDSPAlloc clDSPOp::FIRWork [private]

Definition at line 161 of file DSPOp.hh.

Referenced by FIRFilter().

5.10.4.10 float clDSPOp::fpIIR_C[5] [private]

Definition at line 163 of file DSPOp.hh.

Referenced by IIRFilter(), and IIRInitialize().

5.10.4.11 float clDSPOp::fpIIR_X[3] [private]

Definition at line 164 of file DSPOp.hh.

Referenced by IIRClear(), and IIRFilter().

5.10.4.12 float clDSPOp::fpIIR_Y[2] [private]

Definition at line 165 of file DSPOp.hh.

Referenced by IIRClear(), and IIRFilter().

5.10.4.13 double clDSPOp::dpIIR_C[5] [private]

Definition at line 166 of file DSPOp.hh.

Referenced by IIRFilter(), and IIRInitialize().

5.10.4.14 double clDSPOp::dpIIR_X[3] [private]

Definition at line 167 of file DSPOp.hh.

Referenced by IIRClear(), and IIRFilter().

5.10.4.15 double clDSPOp::dpIIR_Y[2] [private]

Definition at line 168 of file DSPOp.hh.

Referenced by IIRClear(), and IIRFilter().

5.10.4.16 bool clDSPOp::bFFTInitialized [private]

Definition at line 170 of file DSPOp.hh.

Referenced by clDSPOp(), DCTInitialize(), DCTUninitialize(), FFTInitialize(), FFTUninitialize(), and ~clDSPOp().

5.10.4.17 bool clDSPOp::bRealTransform [private]

Definition at line 171 of file DSPOp.hh.

Referenced by FFTInitialize(), and FFTUninitialize().

5.10.4.18 long clDSPOp::lFFTLength [private]

Definition at line 172 of file DSPOp.hh.

Referenced by DCTi(), DCTInitialize(), DCTo(), FFTi(), FFTInitialize(), FFTo(), IDCTi(), IDCTo(), and IFFTo().

5.10.4.19 float clDSPOp::fFFTScale [private]

Definition at line 173 of file DSPOp.hh.

Referenced by DCTInitialize(), FFTi(), FFTInitialize(), FFTo(), IDCTi(), and IDCTo().

5.10.4.20 double clDSPOp::dFFTScale [private]

Definition at line 174 of file DSPOp.hh.

Referenced by DCTInitialize(), FFTi(), FFTInitialize(), FFTo(), IDCTi(), and IDCTo().

5.10.4.21 long* clDSPOp::lpSBitRevWork [private]

Definition at line 175 of file DSPOp.hh.

Referenced by clDSPOp(), DCTi(), DCTInitialize(), DCTo(), FFTi(), FFTInitialize(), FFTo(), IDCTi(), IDCTo(), and IFFTo().

5.10.4.22 long* clDSPOp::lpDBitRevWork [private]

Definition at line 176 of file DSPOp.hh.

Referenced by clDSPOp(), DCTi(), DCTInitialize(), DCTo(), FFTi(), FFTInitialize(), FFTo(), IDCTi(), IDCTo(), and IFFTo().

5.10.4.23 float* clDSPOp::fpCosSinTable [private]

Definition at line 177 of file DSPOp.hh.

Referenced by clDSPOp(), DCTi(), DCTInitialize(), DCTo(), FFTi(), FFTInitialize(), FFTo(), IDCTi(), IDCTo(), and IFFTo().

5.10.4.24 double* clDSPOp::dpCosSinTable [private]

Definition at line 178 of file DSPOp.hh.

Referenced by clDSPOp(), DCTi(), DCTInitialize(), DCTo(), FFTi(), FFTInitialize(), FFTo(), IDCTi(), IDCTo(), and IFFTo().

5.10.4.25 void* clDSPOp::vpSTfrm [private]

Definition at line 179 of file DSPOp.hh.

Referenced by FFTi(), FFTInitialize(), FFTo(), FFTUninitialize(), and IFFTo().

5.10.4.26 void* clDSPOp::vpDTfrm [private]

Definition at line 180 of file DSPOp.hh.

Referenced by FFTi(), FFTInitialize(), FFTo(), FFTUninitialize(), and IFFTo().

5.10.4.27 clDSPAlloc clDSPOp::SBitRevWork [private]

Definition at line 181 of file DSPOp.hh.

Referenced by DCTInitialize(), DCTUninitialize(), FFTi(), FFTInitialize(), FFTo(), FFTUninitialize(), and IFFTo().

5.10.4.28 clDSPAlloc clDSPOp::DBitRevWork [private]

Definition at line 182 of file DSPOp.hh.

Referenced by DCTInitialize(), DCTUninitialize(), FFTi(), FFTInitialize(), FFTo(), FFTUninitialize(), and IFFTo().

5.10.4.29 clDSPAlloc clDSPOp::SCosSinTable [private]

Definition at line 183 of file DSPOp.hh.

Referenced by DCTInitialize(), DCTUninitialize(), FFTInitialize(), and FFTUninitialize().

5.10.4.30 clDSPAlloc clDSPOp::DCosSinTable [private]

Definition at line 184 of file DSPOp.hh.

Referenced by DCTInitialize(), DCTUninitialize(), FFTInitialize(), and FFTUninitialize().

5.10.4.31 clDSPAlloc clDSPOp::FFTBuf [private]

Definition at line 185 of file DSPOp.hh.

Referenced by DCTInitialize(), DCTUninitialize(), FFTInitialize(), FFTo(), FFTUninitialize(), and IFFTo().

5.10.4.32 clTransformS clDSPOp::Tfrm [private]

Definition at line 197 of file DSPOp.hh.

Referenced by DCTi(), DCTInitialize(), DCTo(), FFTi(), FFTInitialize(), FFTo(), IDCTi(), IDCTo(), and IFFTo().

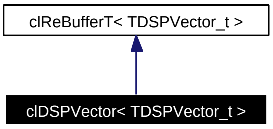
The documentation for this class was generated from the following files:

- DSPOp.hh
- DSPOp.cc

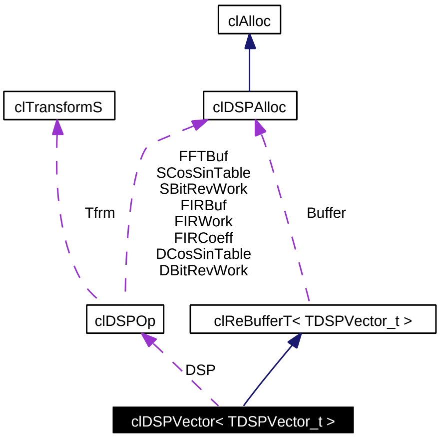
5.11 clDSPVector< TDSPVector_t > Class Template Reference

`#include <DSPVector.hh>`

Inheritance diagram for clDSPVector< TDSPVector_t >:



Collaboration diagram for clDSPVector< TDSPVector_t >:



Public Member Functions

- long Size ()
- void SetSize (long lNewSize)

    Set number of samples in FIFO.

- 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 ()
- clDSPVector & Normalize ()
- clDSPVector & Reverse ()
- clDSPVector & Reverse (clDSPVector &Src)
- clDSPVector & Scale ()
- clDSPVector & Scale (clDSPVector &Src)
- clDSPVector & Scale01 ()
- clDSPVector & Scale01 (clDSPVector &Src)
- clDSPVector & Sort ()
- clDSPVector & Sort (clDSPVector &Src)
- TDSPVector_t Sum ()
- TDSPVector_t Conv (clDSPVector &Src)
- clDSPVector & Conv (clDSPVector &Src1, clDSPVector &Src2)
- TDSPVector_t Corr (clDSPVector &Src)
- clDSPVector & Corr (clDSPVector &Src1, clDSPVector &Src2)
- TDSPVector_t AutoCorr ()

- clDSPVector & AutoCorr (clDSPVector &Src)
- TDSPVector_t CrossCorr (clDSPVector &Src)
- TDSPVector_t CrossCorr (clDSPVector &Src, long lDelay)
- clDSPVector & CrossCorr (clDSPVector &Src1, clDSPVector &Src2, long *lpDelays, long lDelayCount)
- TDSPVector_t DotProduct (clDSPVector &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)
- clDSPVector & Convert (const unsigned char *ucpSrc, long lSrcCount)
- clDSPVector & Convert (const signed short *ipSrc, long lSrcCount, bool b12bit=false)
- clDSPVector & Convert (const signed int *ipSrc, long lSrcCount, bool b24bit=false)
- clDSPVector & Convert (const float *fpSrc, long lSrcCount)
- clDSPVector & 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)
- clDSPVector & CartToPolar ()
- clDSPVector & CartToPolar (clDSPVector &Cart)
- void CartToPolar (clDSPVector< float > &Magn, clDSPVector< float > &Phase)
- void CartToPolar (clDSPVector< double > &Magn, clDSPVector< 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 ()
- void DCTInitialize (long lWinSize)
- void DCTUninitialize ()
- clDSPVector & DCT ()
- clDSPVector & DCT (clDSPVector< float > &Src)
- clDSPVector & DCT (clDSPVector< double > &Src)
- clDSPVector & IDCT ()
- clDSPVector & IDCT (clDSPVector< float > &Src)
- clDSPVector & IDCT (clDSPVector< double > &Src)

Protected Member Functions

- TDSPVector_t ∗ GetPtr ()

Private Attributes

- long lFFTSize
- clDSPOp DSP

template<class TDSPVector_t> class clDSPVector< TDSPVector_t >

5.11.1 Constructor & Destructor Documentation

5.11.1.1 template<class TDSPVector_t> clDSPVector< TDSPVector_t >::clDSPVector () `[inline]`

Definition at line 48 of file DSPVector.hh.

References clDSPVector< TDSPVector_t >::lFFTSize.

5.11.1.2 template<class TDSPVector_t> clDSPVector< TDSPVector_t >::clDSPVector (const clDSPVector< TDSPVector_t > & *CopySrc*) `[inline]`

Definition at line 52 of file DSPVector.hh.

References clDSPVector< TDSPVector_t >::lFFTSize.

5.11.1.3 template<class TDSPVector_t> clDSPVector< TDSPVector_t >::clDSPVector (long *lNewSize*) `[inline]`

Definition at line 57 of file DSPVector.hh.

References clDSPVector< TDSPVector_t >::lFFTSize.

5.11.1.4 template<class TDSPVector_t> clDSPVector< TDSPVector_t >::clDSPVector (const TDSPVector_t ∗ *fpSrcData*, long *lSrcCount*) `[inline]`

Definition at line 62 of file DSPVector.hh.

References clDSPVector< TDSPVector_t >::lFFTSize.

5.11.1.5 template<class TDSPVector_t> clDSPVector< TDSPVector_t >::~clDSPVector () [inline]

Definition at line 67 of file DSPVector.hh.

5.11.2 Member Function Documentation

5.11.2.1 template<class TDSPVector_t> TDSPVector_t* clDSPVector< TDSPVector_t >::GetPtr () [inline, protected]

Reimplemented from clReBufferT< TDSPVector_t >.

Definition at line 40 of file DSPVector.hh.

References clReBufferT< TReBuffer_t >::GetPtr().

Referenced by clDSPVector< TDSPVector_t >::Ptr().

5.11.2.2 template<class TDSPVector_t> long clDSPVector< TDSPVector_t >::Size () [inline]

Definition at line 43 of file DSPVector.hh.

References clReBufferT< TReBuffer_t >::Size().

Referenced by clDSPVector< TDSPVector_t >::Abs(), clDSPVector< TDSPVector_t >::AutoCorr(), clDSPVector< TDSPVector_t >::CartToPolar(), clDSPVector< TDSPVector_t >::Clip(), clDSPVector< TDSPVector_t >::ClipZero(), clDSPVector< TDSPVector_t >::Convert(), clDSPVector< TDSPVector_t >::Decimate(), clDSPVector< TDSPVector_t >::DecimateAvg(), clDSPVector< TDSPVector_t >::Div1x(), clDSPVector< TDSPVector_t >::Energy(), clDSPVector< TDSPVector_t >::Extract(), clDSPVector< TDSPVector_t >::FFTWConvert(), clDSPVector< TDSPVector_t >::FIRAllocate(), clDSPVector< TDSPVector_t >::FIRFilter(), clDSPVector< TDSPVector_t >::Interpolate(), clDSPVector< TDSPVector_t >::InterpolateAvg(), clDSPVector< TDSPVector_t >::Magnitude(), clDSPVector< TDSPVector_t >::Mean(), clDSPVector< TDSPVector_t >::Median(), clDSPVector< TDSPVector_t >::MinMax(), clDSPVector< TDSPVector_t >::Mix(), clDSPVector< TDSPVector_t >::MulAdd(), clDSPVector< TDSPVector_t >::Negate(), clDSPVector< TDSPVector_t >::Normalize(), clDSPVector< TDSPVector_t >::operator *(), clDSPVector< TDSPVector_t >::operator *=(), clDSPVector< TDSPVector_t >::operator+(), clDSPVector< TDSPVector_t >::operator+=(), clDSPVector< TDSPVector_t >::operator-(), clDSPVector< TDSPVector_t >::operator-=(), clDSPVector< TDSPVector_t >::operator/(), clDSPVector< TDSPVector_t >::operator/=(), clDSPVector< TDSPVector_t >::Pack(), clDSPVector< TDSPVector_t >::PeakLevel(), clDSPVector< TDSPVector_t >::Phase(), clDSPVector< TDSPVector_t >::PolarToCart(), clDSPVector< TDSPVector_t >::Power(), clDSPVector< TDSPVector_t >::PowerPhase(), clDSPVector< TDSPVector_t >::Resample(), clDSPVector< TDSPVector_t

>::ResampleAvg(), clDSPVector< TDSPVector_t >::Reverse(), clDSPVector< TDSPVector_t >::RMS(), clDSPVector< TDSPVector_t >::Scale(), clDSPVector< TDSPVector_t >::Scale01(), clDSPVector< TDSPVector_t >::Set(), clDSPVector< TDSPVector_t >::Sort(), clDSPVector< TDSPVector_t >::Sqrt(), clDSPVector< TDSPVector_t >::Square(), clDSPVector< TDSPVector_t >::StdDev(), clDSPVector< TDSPVector_t >::Sum(), clDSPVector< TDSPVector_t >::Variance(), clDSPVector< TDSPVector_t >::WinBartlett(), clDSPVector< TDSPVector_t >::WinBlackman(), clDSPVector< TDSPVector_t >::WinBlackmanHarris(), clDSPVector< TDSPVector_t >::WinCosTapered(), clDSPVector< TDSPVector_t >::WinDolphChebyshev(), clDSPVector< TDSPVector_t >::WinExactBlackman(), clDSPVector< TDSPVector_t >::WinExp(), clDSPVector< TDSPVector_t >::WinFlatTop(), clDSPVector< TDSPVector_t >::WinGenericCos(), clDSPVector< TDSPVector_t >::WinHamming(), clDSPVector< TDSPVector_t >::WinHanning(), clDSPVector< TDSPVector_t >::WinKaiser(), clDSPVector< TDSPVector_t >::WinKaiserBessel(), clDSPVector< TDSPVector_t >::WinTukey(), and clDSPVector< TDSPVector_t >::Zero().

5.11.2.3 template<class TDSPVector_t> void clDSPVector< TDSPVector_t >::SetSize (long *lNewSize*) [inline]

Set number of samples in FIFO.

Note:

This is destructive resize.

Reimplemented from clReBufferT< TDSPVector_t >.

Definition at line 45 of file DSPVector.hh.

References clReBufferT< TReBuffer_t >::SetSize().

Referenced by clDSPVector< TDSPVector_t >::AutoCorr(), clDSPVector< TDSPVector_t >::CartToPolar(), clDSPVector< TDSPVector_t >::Conv(), clDSPVector< TDSPVector_t >::Convert(), clDSPVector< TDSPVector_t >::Corr(), clDSPVector< TDSPVector_t >::CrossCorr(), clDSPVector< TDSPVector_t >::DCT(), clDSPVector< TDSPVector_t >::Decimate(), clDSPVector< TDSPVector_t >::DecimateAvg(), clDSPVector< TDSPVector_t >::Extract(), clDSPVector< TDSPVector_t >::FFT(), clDSPVector< TDSPVector_t >::FIRFilter(), clDSPVector< TDSPVector_t >::IDCT(), clDSPVector< TDSPVector_t >::IFFT(), clDSPVector< TDSPVector_t >::Interpolate(), clDSPVector< TDSPVector_t >::InterpolateAvg(), clDSPVector< TDSPVector_t >::Magnitude(), clDSPVector< TDSPVector_t >::Mix(), clDSPVector< TDSPVector_t >::Phase(), clDSPVector< TDSPVector_t >::PolarToCart(), clDSPVector< TDSPVector_t >::Power(), clDSPVector< TDSPVector_t >::PowerPhase(), clDSPVector< TDSPVector_t >::Reverse(), clDSPVector< TDSPVector_t >::Scale(), clDSPVector< TDSPVector_t >::Scale01(), clDSPVector< TDSPVector_t >::Set(), clDSPVector< TDSPVector_t >::WinBartlett(), clDSPVector< TDSPVector_t >::WinBlackman(), clDSPVector< TDSPVector_t >::WinBlackmanHarris(), clDSPVector< TDSPVector_t >::WinCosTapered(), clDSPVector< TDSPVector_t >::WinDolphChebyshev(), clDSPVector< TDSPVector_t >::WinExactBlackman(), clDSPVector< TDSPVector_t >::WinExp(), clDSPVector< TDSPVector_t >::WinFlatTop(), clDSPVector< TDSPVector_t >::WinGenericCos(), clDSPVector< TDSPVector_t >::WinHamming(), clDSPVector< TDSPVector_t >::WinHanning(), clDSPVector< TDSPVector_t >::WinKaiser(), clDSPVector< TDSPVector_t >::WinKaiserBessel(), and clDSPVector< TDSPVector_t >::WinTukey().

5.11.2.4 template<class TDSPVector_t> TDSPVector_t* clDSPVector< TDSPVector_t >::Ptr () [inline]

Definition at line 68 of file DSPVector.hh.

References clDSPVector< TDSPVector_t >::GetPtr().

Referenced by clDSPVector< TDSPVector_t >::Abs(), clDSPVector< TDSPVector_t >::AutoCorr(), clDSPVector< TDSPVector_t >::CartToPolar(), clDSPVector< TDSPVector_t >::Clip(), clDSPVector< TDSPVector_t >::ClipZero(), clDSPVector< TDSPVector_t >::Conv(), clDSPVector< TDSPVector_t >::Convert(), clDSPVector< TDSPVector_t >::Corr(), clDSPVector< TDSPVector_t >::CrossCorr(), clDSPVector< TDSPVector_t >::DCT(), clDSPVector< TDSPVector_t >::Decimate(), clDSPVector< TDSPVector_t >::DecimateAvg(), clDSPVector< TDSPVector_t >::Div1x(), clDSPVector< TDSPVector_t >::DotProduct(), clDSPVector< TDSPVector_t >::Energy(), clDSPVector< TDSPVector_t >::Extract(), clDSPVector< TDSPVector_t >::FFT(), clDSPVector< TDSPVector_t >::FFTWConvert(), clDSPVector< TDSPVector_t >::FIRAllocate(), clDSPVector< TDSPVector_t >::FIRFilter(), clDSPVector< TDSPVector_t >::IDCT(), clDSPVector< TDSPVector_t >::IFFT(), clDSPVector< TDSPVector_t >::Interpolate(), clDSPVector< TDSPVector_t >::InterpolateAvg(), clDSPVector< TDSPVector_t >::Magnitude(), clDSPVector< TDSPVector_t >::Mean(), clDSPVector< TDSPVector_t >::Median(), clDSPVector< TDSPVector_t >::MinMax(), clDSPVector< TDSPVector_t >::Mix(), clDSPVector< TDSPVector_t >::MulAdd(), clDSPVector< TDSPVector_t >::MulC(), clDSPVector< TDSPVector_t >::Negate(), clDSPVector< TDSPVector_t >::Normalize(), clDSPVector< TDSPVector_t >::operator *(), clDSPVector< TDSPVector_t >::operator *=(), clDSPVector< TDSPVector_t >::operator+(), clDSPVector< TDSPVector_t >::operator+=(), clDSPVector< TDSPVector_t >::operator-(), clDSPVector< TDSPVector_t >::operator-=(), clDSPVector< TDSPVector_t >::operator/(), clDSPVector< TDSPVector_t >::operator/=(), clDSPVector< TDSPVector_t >::Pack(), clDSPVector< TDSPVector_t >::PeakLevel(), clDSPVector< TDSPVector_t >::Phase(), clDSPVector< TDSPVector_t >::PolarToCart(), clDSPVector< TDSPVector_t >::Power(), clDSPVector< TDSPVector_t >::PowerPhase(), clDSPVector< TDSPVector_t >::Resample(), clDSPVector< TDSPVector_t >::ResampleAvg(), clDSPVector< TDSPVector_t >::Reverse(), clDSPVector< TDSPVector_t >::RMS(), clDSPVector< TDSPVector_t >::Scale(), clDSPVector< TDSPVector_t >::Scale01(), clDSPVector< TDSPVector_t >::Set(), clDSPVector< TDSPVector_t >::Sort(), clDSPVector< TDSPVector_t >::Sqrt(), clDSPVector< TDSPVector_t >::Square(), clDSPVector< TDSPVector_t >::StdDev(), clDSPVector< TDSPVector_t >::Sum(), clDSPVector< TDSPVector_t >::Variance(), clDSPVector< TDSPVector_t >::WinBartlett(), clDSPVector< TDSPVector_t >::WinBlackman(), clDSPVector< TDSPVector_t >::WinBlackmanHarris(), clDSPVector< TDSPVector_t >::WinCosTapered(), clDSPVector< TDSPVector_t

>::WinDolphChebyshev(), clDSPVector< TDSPVector_t >::WinExactBlackman(), clDSPVector< TDSPVector_t >::WinExp(), clDSPVector< TDSPVector_t >::WinFlatTop(), clDSPVector< TDSPVector_t >::WinGenericCos(), clDSPVector< TDSPVector_t >::WinHamming(), clDSPVector< TDSPVector_t >::WinHanning(), clDSPVector< TDSPVector_t >::WinKaiser(), clDSPVector< TDSPVector_t >::WinKaiserBessel(), clDSPVector< TDSPVector_t >::WinTukey(), and clDSPVector< TDSPVector_t >::Zero().

5.11.2.5 template<class TDSPVector_t> clDSPVector clDSPVector< TDSPVector_t >::operator+ (const TDSPVector_t & *Src*) `[inline]`

Definition at line 73 of file DSPVector.hh.

References clDSPOp::Add(), clDSPVector< TDSPVector_t >::DSP, clDSPVector< TDSPVector_t >::Ptr(), and clDSPVector< TDSPVector_t >::Size().

5.11.2.6 template<class TDSPVector_t> clDSPVector clDSPVector< TDSPVector_t >::operator+ (clDSPVector< TDSPVector_t > & *Src*) `[inline]`

Definition at line 79 of file DSPVector.hh.

References clDSPOp::Add(), clDSPVector< TDSPVector_t >::DSP, DSPV_SHORTER, and clDSPVector< TDSPVector_t >::Ptr().

5.11.2.7 template<class TDSPVector_t> clDSPVector clDSPVector< TDSPVector_t >::operator- (const TDSPVector_t & *Src*) `[inline]`

Definition at line 85 of file DSPVector.hh.

References clDSPVector< TDSPVector_t >::DSP, clDSPVector< TDSPVector_t >::Ptr(), clDSPVector< TDSPVector_t >::Size(), and clDSPOp::Sub().

5.11.2.8 template<class TDSPVector_t> clDSPVector clDSPVector< TDSPVector_t >::operator- (clDSPVector< TDSPVector_t > & *Src*) `[inline]`

Definition at line 91 of file DSPVector.hh.

References clDSPVector< TDSPVector_t >::DSP, DSPV_SHORTER, clDSPVector< TDSPVector_t >::Ptr(), and clDSPOp::Sub().

5.11.2.9 template<class TDSPVector_t> clDSPVector clDSPVector< TDSPVector_t >::operator * (const TDSPVector_t & *Src*) `[inline]`

Definition at line 97 of file DSPVector.hh.

References clDSPVector< TDSPVector_t >::DSP, clDSPOp::Mul(), clDSPVector< TDSPVector_t >::Ptr(), and clDSPVector< TDSPVector_t >::Size().

5.11.2.10 template<class TDSPVector_t> clDSPVector clDSPVector< TDSPVector_t >::operator * (clDSPVector< TDSPVector_t > & *Src*) [inline]

Definition at line 103 of file DSPVector.hh.

References clDSPVector< TDSPVector_t >::DSP, DSPV_SHORTER, clDSPOp::Mul(), and clDSPVector< TDSPVector_t >::Ptr().

5.11.2.11 template<class TDSPVector_t> clDSPVector clDSPVector< TDSPVector_t >::operator/ (const TDSPVector_t & *Src*) [inline]

Definition at line 109 of file DSPVector.hh.

References clDSPOp::Div(), clDSPVector< TDSPVector_t >::DSP, clDSPVector< TDSPVector_t >::Ptr(), and clDSPVector< TDSPVector_t >::Size().

5.11.2.12 template<class TDSPVector_t> clDSPVector clDSPVector< TDSPVector_t >::operator/ (clDSPVector< TDSPVector_t > & *Src*) [inline]

Definition at line 115 of file DSPVector.hh.

References clDSPOp::Div(), clDSPVector< TDSPVector_t >::DSP, DSPV_SHORTER, and clDSPVector< TDSPVector_t >::Ptr().

5.11.2.13 template<class TDSPVector_t> clDSPVector& clDSPVector< TDSPVector_t >::operator+= (const TDSPVector_t & *Src*) [inline]

Definition at line 121 of file DSPVector.hh.

References clDSPOp::Add(), clDSPVector< TDSPVector_t >::DSP, clDSPVector< TDSPVector_t >::Ptr(), and clDSPVector< TDSPVector_t >::Size().

5.11.2.14 template<class TDSPVector_t> clDSPVector& clDSPVector< TDSPVector_t >::operator+= (clDSPVector< TDSPVector_t > & *Src*) [inline]

Definition at line 126 of file DSPVector.hh.

References clDSPOp::Add(), clDSPVector< TDSPVector_t >::DSP, DSPV_SHORTER, and clDSPVector< TDSPVector_t >::Ptr().

5.11.2.15 template<class TDSPVector_t> clDSPVector& clDSPVector< TDSPVector_t >::operator-= (const TDSPVector_t & *Src*) [inline]

Definition at line 131 of file DSPVector.hh.

References clDSPVector< TDSPVector_t >::DSP, clDSPVector< TDSPVector_-t >::Ptr(), clDSPVector< TDSPVector_t >::Size(), and clDSPOp::Sub().

5.11.2.16 template<class TDSPVector_t> clDSPVector& clDSPVector< TDSPVector_t >::operator-= (clDSPVector< TDSPVector_t > & *Src*) `[inline]`

Definition at line 136 of file DSPVector.hh.

References clDSPVector< TDSPVector_t >::DSP, DSPV_SHORTER, clDSPVector< TDSPVector_t >::Ptr(), and clDSPOp::Sub().

5.11.2.17 template<class TDSPVector_t> clDSPVector& clDSPVector< TDSPVector_t >::operator *= (const TDSPVector_t & *Src*) `[inline]`

Definition at line 141 of file DSPVector.hh.

References clDSPVector< TDSPVector_t >::DSP, clDSPOp::Mul(), clDSPVector< TDSPVector_t >::Ptr(), and clDSPVector< TDSPVector_t >::Size().

5.11.2.18 template<class TDSPVector_t> clDSPVector& clDSPVector< TDSPVector_t >::operator *= (clDSPVector< TDSPVector_t > & *Src*) `[inline]`

Definition at line 146 of file DSPVector.hh.

References clDSPVector< TDSPVector_t >::DSP, DSPV_SHORTER, cl-DSPOp::Mul(), and clDSPVector< TDSPVector_t >::Ptr().

5.11.2.19 template<class TDSPVector_t> clDSPVector& clDSPVector< TDSPVector_t >::operator/= (const TDSPVector_t & *Src*) `[inline]`

Definition at line 151 of file DSPVector.hh.

References clDSPOp::Div(), clDSPVector< TDSPVector_t >::DSP, clDSPVector< TDSPVector_t >::Ptr(), and clDSPVector< TDSPVector_t >::Size().

5.11.2.20 template<class TDSPVector_t> clDSPVector& clDSPVector< TDSPVector_t >::operator/= (clDSPVector< TDSPVector_t > & *Src*) `[inline]`

Definition at line 156 of file DSPVector.hh.

References clDSPOp::Div(), clDSPVector< TDSPVector_t >::DSP, DSPV_-SHORTER, and clDSPVector< TDSPVector_t >::Ptr().

5.11.2.21 template<class TDSPVector_t> clDSPVector& clDSPVector< TDSPVector_t >::Zero () [inline]

Definition at line 162 of file DSPVector.hh.

References clDSPVector< TDSPVector_t >::DSP, clDSPVector< TDSPVector_t >::Ptr(), clDSPVector< TDSPVector_t >::Size(), and clDSPOp::Zero().

5.11.2.22 template<class TDSPVector_t> clDSPVector& clDSPVector< TDSPVector_t >::Set (const TDSPVector_t & *Src*) [inline]

Definition at line 167 of file DSPVector.hh.

References clDSPVector< TDSPVector_t >::DSP, clDSPVector< TDSPVector_t >::Ptr(), clDSPOp::Set(), and clDSPVector< TDSPVector_t >::Size().

5.11.2.23 template<class TDSPVector_t> clDSPVector& clDSPVector< TDSPVector_t >::Set (const TDSPVector_t * *Src*, long *lSrcCount*) [inline]

Definition at line 172 of file DSPVector.hh.

References clDSPVector< TDSPVector_t >::DSP, clDSPVector< TDSPVector_t >::Ptr(), clDSPOp::Set(), and clDSPVector< TDSPVector_t >::SetSize().

5.11.2.24 template<class TDSPVector_t> clDSPVector& clDSPVector< TDSPVector_t >::Set (const TDSPVector_t & *Src*, long *lStartIdx*, long *lSetCount*) [inline]

Definition at line 178 of file DSPVector.hh.

References clDSPVector< TDSPVector_t >::DSP, clDSPVector< TDSPVector_t >::Ptr(), clDSPOp::Set(), and clDSPVector< TDSPVector_t >::Size().

5.11.2.25 template<class TDSPVector_t> clDSPVector& clDSPVector< TDSPVector_t >::Clip (const TDSPVector_t & *Src*) [inline]

Definition at line 184 of file DSPVector.hh.

References clDSPOp::Clip(), clDSPVector< TDSPVector_t >::DSP, clDSPVector< TDSPVector_t >::Ptr(), and clDSPVector< TDSPVector_t >::Size().

5.11.2.26 template<class TDSPVector_t> clDSPVector& clDSPVector< TDSPVector_t >::Clip (const TDSPVector_t & *Src1*, const TDSPVector_t & *Src2*) [inline]

Definition at line 189 of file DSPVector.hh.

References clDSPOp::Clip(), clDSPVector< TDSPVector_t >::DSP, clDSPVector< TDSPVector_t >::Ptr(), and clDSPVector< TDSPVector_t >::Size().

5.11.2.27 template<class TDSPVector_t> clDSPVector& clDSPVector< TDSPVector_t >::ClipZero () [inline]

Definition at line 195 of file DSPVector.hh.

References clDSPOp::ClipZero(), clDSPVector< TDSPVector_t >::DSP, clDSPVector< TDSPVector_t >::Ptr(), and clDSPVector< TDSPVector_t >::Size().

5.11.2.28 template<class TDSPVector_t> clDSPVector& clDSPVector< TDSPVector_t >::MulC (clDSPVector< TDSPVector_t > & *Src*) [inline]

Definition at line 201 of file DSPVector.hh.

References clDSPVector< TDSPVector_t >::DSP, DSPV_SHORTER, clDSPOp::MulC(), and clDSPVector< TDSPVector_t >::Ptr().

5.11.2.29 template<class TDSPVector_t> clDSPVector& clDSPVector< TDSPVector_t >::MulC (clDSPVector< TDSPVector_t > & *Src1*, clDSPVector< TDSPVector_t > & *Src2*) [inline]

Definition at line 206 of file DSPVector.hh.

References clDSPVector< TDSPVector_t >::DSP, DSPV_SHORTER, clDSPOp::MulC(), and clDSPVector< TDSPVector_t >::Ptr().

5.11.2.30 template<class TDSPVector_t> clDSPVector& clDSPVector< TDSPVector_t >::Div1x () [inline]

Definition at line 212 of file DSPVector.hh.

References clDSPOp::Div1x(), clDSPVector< TDSPVector_t >::DSP, clDSPVector< TDSPVector_t >::Ptr(), and clDSPVector< TDSPVector_t >::Size().

5.11.2.31 template<class TDSPVector_t> clDSPVector& clDSPVector< TDSPVector_t >::MulAdd (TDSPVector_t *fMul*, TDSPVector_t *fAdd*) [inline]

Definition at line 217 of file DSPVector.hh.

References clDSPVector< TDSPVector_t >::DSP, clDSPOp::MulAdd(), clDSPVector< TDSPVector_t >::Ptr(), and clDSPVector< TDSPVector_t >::Size().

5.11.2.32 template<class TDSPVector_t> clDSPVector& clDSPVector< TDSPVector_t >::Square () [inline]

Definition at line 221 of file DSPVector.hh.

References clDSPVector< TDSPVector_t >::DSP, clDSPVector< TDSPVector_t >::Ptr(), clDSPVector< TDSPVector_t >::Size(), and clDSPOp::Square().

5.11.2.33 template<class TDSPVector_t> clDSPVector& clDSPVector< TDSPVector_t >::Abs () [inline]

Definition at line 226 of file DSPVector.hh.

References clDSPOp::Abs(), clDSPVector< TDSPVector_t >::DSP, clDSPVector< TDSPVector_t >::Ptr(), and clDSPVector< TDSPVector_t >::Size().

5.11.2.34 template<class TDSPVector_t> clDSPVector& clDSPVector< TDSPVector_t >::Sqrt () [inline]

Definition at line 231 of file DSPVector.hh.

References clDSPVector< TDSPVector_t >::DSP, clDSPVector< TDSPVector_t >::Ptr(), clDSPVector< TDSPVector_t >::Size(), and clDSPOp::Sqrt().

5.11.2.35 template<class TDSPVector_t> clDSPVector& clDSPVector< TDSPVector_t >::Negate () [inline]

Definition at line 236 of file DSPVector.hh.

References clDSPVector< TDSPVector_t >::DSP, clDSPOp::Negate(), clDSPVector< TDSPVector_t >::Ptr(), and clDSPVector< TDSPVector_t >::Size().

5.11.2.36 template<class TDSPVector_t> clDSPVector& clDSPVector< TDSPVector_t >::Normalize () [inline]

Definition at line 241 of file DSPVector.hh.

References clDSPVector< TDSPVector_t >::DSP, clDSPOp::Normalize(), clDSPVector< TDSPVector_t >::Ptr(), and clDSPVector< TDSPVector_t >::Size().

5.11.2.37 template<class TDSPVector_t> clDSPVector& clDSPVector< TDSPVector_t >::Reverse () [inline]

Definition at line 246 of file DSPVector.hh.

References clDSPVector< TDSPVector_t >::DSP, clDSPVector< TDSPVector_t >::Ptr(), clDSPOp::Reverse(), and clDSPVector< TDSPVector_t >::Size().

5.11.2.38 template<class TDSPVector_t> clDSPVector& clDSPVector< TDSPVector_t >::Reverse (clDSPVector< TDSPVector_t > & *Src*) [inline]

Definition at line 251 of file DSPVector.hh.

References clDSPVector< TDSPVector_t >::DSP, DSPV_SHORTER, clDSPVector< TDSPVector_t >::Ptr(), clDSPOp::Reverse(), clDSPVector< TDSPVector_t >::SetSize(), and clDSPVector< TDSPVector_t >::Size().

5.11.2.39 template<class TDSPVector_t> clDSPVector& clDSPVector< TDSPVector_t >::Scale () [inline]

Definition at line 257 of file DSPVector.hh.

References clDSPVector< TDSPVector_t >::DSP, clDSPVector< TDSPVector_t >::Ptr(), clDSPOp::Scale(), and clDSPVector< TDSPVector_t >::Size().

5.11.2.40 template<class TDSPVector_t> clDSPVector& clDSPVector< TDSPVector_t >::Scale (clDSPVector< TDSPVector_t > & *Src*) [inline]

Definition at line 262 of file DSPVector.hh.

References clDSPVector< TDSPVector_t >::DSP, DSPV_SHORTER, clDSPVector< TDSPVector_t >::Ptr(), clDSPOp::Scale(), clDSPVector< TDSPVector_t >::SetSize(), and clDSPVector< TDSPVector_t >::Size().

5.11.2.41 template<class TDSPVector_t> clDSPVector& clDSPVector< TDSPVector_t >::Scale01 () [inline]

Definition at line 268 of file DSPVector.hh.

References clDSPVector< TDSPVector_t >::DSP, clDSPVector< TDSPVector_t >::Ptr(), clDSPOp::Scale01(), and clDSPVector< TDSPVector_t >::Size().

5.11.2.42 template<class TDSPVector_t> clDSPVector& clDSPVector< TDSPVector_t >::Scale01 (clDSPVector< TDSPVector_t > & *Src*) [inline]

Definition at line 273 of file DSPVector.hh.

References clDSPVector< TDSPVector_t >::DSP, clDSPVector< TDSPVector_t >::Ptr(), clDSPOp::Scale01(), clDSPVector< TDSPVector_t >::SetSize(), and clDSPVector< TDSPVector_t >::Size().

5.11.2.43 template<class TDSPVector_t> clDSPVector& clDSPVector< TDSPVector_t >::Sort () [inline]

Definition at line 279 of file DSPVector.hh.

References clDSPVector< TDSPVector_t >::DSP, clDSPVector< TDSPVector_t >::Ptr(), clDSPVector< TDSPVector_t >::Size(), and clDSPOp::Sort().

5.11.2.44 template<class TDSPVector_t> clDSPVector& clDSPVector< TDSPVector_t >::Sort (clDSPVector< TDSPVector_t > & *Src*) [inline]

Definition at line 284 of file DSPVector.hh.

References clDSPVector< TDSPVector_t >::DSP, clDSPVector< TDSPVector_t >::Ptr(), clDSPVector< TDSPVector_t >::Size(), and clDSPOp::Sort().

5.11.2.45 template<class TDSPVector_t> TDSPVector_t clDSPVector< TDSPVector_t >::Sum () [inline]

Definition at line 291 of file DSPVector.hh.

References clDSPVector< TDSPVector_t >::DSP, clDSPVector< TDSPVector_t >::Ptr(), clDSPVector< TDSPVector_t >::Size(), and clDSPOp::Sum().

5.11.2.46 template<class TDSPVector_t> TDSPVector_t clDSPVector< TDSPVector_t >::Conv (clDSPVector< TDSPVector_t > & *Src*) [inline]

Definition at line 295 of file DSPVector.hh.

References clDSPOp::Convolve(), clDSPVector< TDSPVector_t >::DSP, DSPV_SHORTER, and clDSPVector< TDSPVector_t >::Ptr().

5.11.2.47 template<class TDSPVector_t> clDSPVector& clDSPVector< TDSPVector_t >::Conv (clDSPVector< TDSPVector_t > & *Src1*, clDSPVector< TDSPVector_t > & *Src2*) [inline]

Definition at line 299 of file DSPVector.hh.

References clDSPOp::Convolve(), clDSPVector< TDSPVector_t >::DSP, DSPV_SHORTER2, clDSPVector< TDSPVector_t >::Ptr(), and clDSPVector< TDSPVector_t >::SetSize().

5.11.2.48 template<class TDSPVector_t> TDSPVector_t clDSPVector< TDSPVector_t >::Corr (clDSPVector< TDSPVector_t > & *Src*) [inline]

Definition at line 306 of file DSPVector.hh.

References clDSPOp::Correlate(), clDSPVector< TDSPVector_t >::DSP, DSPV_-SHORTER, and clDSPVector< TDSPVector_t >::Ptr().

5.11.2.49 template<class TDSPVector_t> clDSPVector& clDSPVector< TDSPVector_t >::Corr (clDSPVector< TDSPVector_t > & *Src1*, clDSPVector< TDSPVector_t > & *Src2*) `[inline]`

Definition at line 310 of file DSPVector.hh.

References clDSPOp::Correlate(), clDSPVector< TDSPVector_t >::DSP, DSPV_SHORTER2, clDSPVector< TDSPVector_t >::Ptr(), and clDSPVector< TDSPVector_t >::SetSize().

5.11.2.50 template<class TDSPVector_t> TDSPVector_t clDSPVector< TDSPVector_t >::AutoCorr () `[inline]`

Definition at line 317 of file DSPVector.hh.

References clDSPOp::AutoCorrelate(), clDSPVector< TDSPVector_t >::DSP, cl-DSPVector< TDSPVector_t >::Ptr(), and clDSPVector< TDSPVector_t >::Size().

5.11.2.51 template<class TDSPVector_t> clDSPVector& clDSPVector< TDSPVector_t >::AutoCorr (clDSPVector< TDSPVector_t > & *Src*) `[inline]`

Definition at line 321 of file DSPVector.hh.

References clDSPOp::AutoCorrelate(), clDSPVector< TDSPVector_t >::DSP, DSPV_SHORTER, clDSPVector< TDSPVector_t >::Ptr(), clDSPVector< TDSPVector_t >::SetSize(), and clDSPVector< TDSPVector_t >::Size().

5.11.2.52 template<class TDSPVector_t> TDSPVector_t clDSPVector< TDSPVector_t >::CrossCorr (clDSPVector< TDSPVector_t > & *Src*) `[inline]`

Definition at line 326 of file DSPVector.hh.

References clDSPOp::CrossCorr(), clDSPVector< TDSPVector_t >::DSP, DSPV_-SHORTER, and clDSPVector< TDSPVector_t >::Ptr().

5.11.2.53 template<class TDSPVector_t> TDSPVector_t clDSPVector< TDSPVector_t >::CrossCorr (clDSPVector< TDSPVector_t > & *Src*, long *lDelay*) `[inline]`

Definition at line 330 of file DSPVector.hh.

References clDSPOp::DelCrossCorr(), clDSPVector< TDSPVector_t >::DSP, DSPV_SHORTER, and clDSPVector< TDSPVector_t >::Ptr().

5.11.2.54 template<class TDSPVector_t> clDSPVector& clDSPVector< TDSPVector_t >::CrossCorr (clDSPVector< TDSPVector_t > & *Src1*, clDSPVector< TDSPVector_t > & *Src2*, long ∗ *lpDelays*, long *lDelayCount*) [inline]

Definition at line 335 of file DSPVector.hh.

References clDSPOp::DelCrossCorr(), clDSPVector< TDSPVector_t >::DSP, DSPV_SHORTER2, clDSPVector< TDSPVector_t >::Ptr(), and clDSPVector< TDSPVector_t >::SetSize().

5.11.2.55 template<class TDSPVector_t> TDSPVector_t clDSPVector< TDSPVector_t >::DotProduct (clDSPVector< TDSPVector_t > & *Src*) [inline]

Definition at line 343 of file DSPVector.hh.

References clDSPOp::DotProduct(), clDSPVector< TDSPVector_t >::DSP, DSPV_-SHORTER, and clDSPVector< TDSPVector_t >::Ptr().

5.11.2.56 template<class TDSPVector_t> TDSPVector_t clDSPVector< TDSPVector_t >::Mean () [inline]

Definition at line 347 of file DSPVector.hh.

References clDSPVector< TDSPVector_t >::DSP, clDSPOp::Mean(), clDSPVector< TDSPVector_t >::Ptr(), and clDSPVector< TDSPVector_t >::Size().

5.11.2.57 template<class TDSPVector_t> TDSPVector_t clDSPVector< TDSPVector_t >::Median () [inline]

Definition at line 351 of file DSPVector.hh.

References clDSPVector< TDSPVector_t >::DSP, clDSPOp::Median(), cl-DSPVector< TDSPVector_t >::Ptr(), and clDSPVector< TDSPVector_t >::Size().

5.11.2.58 template<class TDSPVector_t> TDSPVector_t clDSPVector< TDSPVector_t >::Energy () [inline]

Definition at line 355 of file DSPVector.hh.

References clDSPVector< TDSPVector_t >::DSP, clDSPOp::Energy(), cl-DSPVector< TDSPVector_t >::Ptr(), and clDSPVector< TDSPVector_t >::Size().

5.11.2.59 template<class TDSPVector_t> TDSPVector_t clDSPVector< TDSPVector_t >::RMS () [inline]

Definition at line 359 of file DSPVector.hh.

References clDSPVector< TDSPVector_t >::DSP, clDSPVector< TDSPVector_-t >::Ptr(), clDSPOp::RMS(), and clDSPVector< TDSPVector_t >::Size().

5.11.2.60 template<class TDSPVector_t> TDSPVector_t clDSPVector< TDSPVector_t >::PeakLevel () [inline]

Definition at line 363 of file DSPVector.hh.

References clDSPVector< TDSPVector_t >::DSP, clDSPOp::PeakLevel(), cl-DSPVector< TDSPVector_t >::Ptr(), and clDSPVector< TDSPVector_t >::Size().

5.11.2.61 template<class TDSPVector_t> void clDSPVector< TDSPVector_t >::MinMax (TDSPVector_t & *fMin*, TDSPVector_t & *fMax*) [inline]

Definition at line 367 of file DSPVector.hh.

References clDSPVector< TDSPVector_t >::DSP, clDSPOp::MinMax(), cl-DSPVector< TDSPVector_t >::Ptr(), and clDSPVector< TDSPVector_t >::Size().

5.11.2.62 template<class TDSPVector_t> void clDSPVector< TDSPVector_t >::StdDev (TDSPVector_t & *fStdDev*, TDSPVector_t & *fMean*) [inline]

Definition at line 371 of file DSPVector.hh.

References clDSPVector< TDSPVector_t >::DSP, clDSPVector< TDSPVector_-t >::Ptr(), clDSPVector< TDSPVector_t >::Size(), and clDSPOp::StdDev().

5.11.2.63 template<class TDSPVector_t> void clDSPVector< TDSPVector_t >::Variance (TDSPVector_t & *fVariance*, TDSPVector_t & *fMean*) [inline]

Definition at line 375 of file DSPVector.hh.

References clDSPVector< TDSPVector_t >::DSP, clDSPVector< TDSPVector_-t >::Ptr(), clDSPVector< TDSPVector_t >::Size(), and clDSPOp::Variance().

5.11.2.64 template<class TDSPVector_t> clDSPVector& clDSPVector< TDSPVector_t >::Convert (const unsigned char * *ucpSrc*, long *lSrcCount*) [inline]

Definition at line 380 of file DSPVector.hh.

References clDSPOp::Convert(), clDSPVector< TDSPVector_t >::DSP, cl-DSPVector< TDSPVector_t >::Ptr(), and clDSPVector< TDSPVector_t >::SetSize().

5.11.2.65 template<class TDSPVector_t> clDSPVector& clDSPVector< TDSPVector_t >::Convert (const signed short * *ipSrc*, long *lSrcCount*, bool *b12bit* = false) [inline]

Definition at line 387 of file DSPVector.hh.

References clDSPOp::Convert(), clDSPVector< TDSPVector_t >::DSP, clDSPVector< TDSPVector_t >::Ptr(), and clDSPVector< TDSPVector_t >::SetSize().

5.11.2.66 template<class TDSPVector_t> clDSPVector& clDSPVector< TDSPVector_t >::Convert (const signed int * *ipSrc*, long *lSrcCount*, bool *b24bit* = false) [inline]

Definition at line 394 of file DSPVector.hh.

References clDSPOp::Convert(), clDSPVector< TDSPVector_t >::DSP, clDSPVector< TDSPVector_t >::Ptr(), and clDSPVector< TDSPVector_t >::SetSize().

5.11.2.67 template<class TDSPVector_t> clDSPVector& clDSPVector< TDSPVector_t >::Convert (const float * *fpSrc*, long *lSrcCount*) [inline]

Definition at line 401 of file DSPVector.hh.

References clDSPOp::Convert(), clDSPVector< TDSPVector_t >::DSP, clDSPVector< TDSPVector_t >::Ptr(), and clDSPVector< TDSPVector_t >::SetSize().

5.11.2.68 template<class TDSPVector_t> clDSPVector& clDSPVector< TDSPVector_t >::Convert (const double * *dpSrc*, long *lSrcCount*) [inline]

Definition at line 407 of file DSPVector.hh.

References clDSPOp::Convert(), clDSPVector< TDSPVector_t >::DSP, clDSPVector< TDSPVector_t >::Ptr(), and clDSPVector< TDSPVector_t >::SetSize().

5.11.2.69 template<class TDSPVector_t> void clDSPVector< TDSPVector_t >::Convert (unsigned char * *ucpDest*) [inline]

Definition at line 413 of file DSPVector.hh.

References clDSPOp::Convert(), clDSPVector< TDSPVector_t >::DSP, clDSPVector< TDSPVector_t >::Ptr(), and clDSPVector< TDSPVector_t >::Size().

5.11.2.70 template<class TDSPVector_t> void clDSPVector< TDSPVector_t >::Convert (signed short * *ipDest*, bool *b12bit* = false) [inline]

Definition at line 417 of file DSPVector.hh.

References clDSPOp::Convert(), clDSPVector< TDSPVector_t >::DSP, clDSPVector< TDSPVector_t >::Ptr(), and clDSPVector< TDSPVector_t >::Size().

5.11.2.71 template<class TDSPVector_t> void clDSPVector< TDSPVector_t >::Convert (signed int * *ipDest*, bool *b24bit* = false) [inline]

Definition at line 421 of file DSPVector.hh.

References clDSPOp::Convert(), clDSPVector< TDSPVector_t >::DSP, clDSPVector< TDSPVector_t >::Ptr(), and clDSPVector< TDSPVector_t >::Size().

5.11.2.72 template<class TDSPVector_t> void clDSPVector< TDSPVector_t >::Convert (float * *fpDest*) [inline]

Definition at line 425 of file DSPVector.hh.

References clDSPOp::Convert(), clDSPVector< TDSPVector_t >::DSP, clDSPVector< TDSPVector_t >::Ptr(), and clDSPVector< TDSPVector_t >::Size().

5.11.2.73 template<class TDSPVector_t> void clDSPVector< TDSPVector_t >::Convert (double * *dpDest*) [inline]

Definition at line 429 of file DSPVector.hh.

References clDSPOp::Convert(), clDSPVector< TDSPVector_t >::DSP, clDSPVector< TDSPVector_t >::Ptr(), and clDSPVector< TDSPVector_t >::Size().

5.11.2.74 template<class TDSPVector_t> clDSPVector& clDSPVector< TDSPVector_t >::CartToPolar () [inline]

Definition at line 433 of file DSPVector.hh.

References clDSPOp::CartToPolar(), clDSPVector< TDSPVector_t >::DSP, clDSPVector< TDSPVector_t >::Ptr(), and clDSPVector< TDSPVector_t >::Size().

5.11.2.75 template<class TDSPVector_t> clDSPVector& clDSPVector< TDSPVector_t >::CartToPolar (clDSPVector< TDSPVector_t > & *Cart*) [inline]

Definition at line 438 of file DSPVector.hh.

References clDSPOp::CartToPolar(), clDSPVector< TDSPVector_t >::DSP, clDSPVector< TDSPVector_t >::Ptr(), clDSPVector< TDSPVector_t >::SetSize(), and clDSPVector< TDSPVector_t >::Size().

5.11.2.76 template<class TDSPVector_t> void clDSPVector< TDSPVector_t >::CartToPolar (clDSPVector< float > & *Magn*, clDSPVector< float > & *Phase*) [inline]

Definition at line 444 of file DSPVector.hh.

References clDSPOp::CartToPolar(), clDSPVector< TDSPVector_t >::DSP, clDSPVector< TDSPVector_t >::Ptr(), clDSPVector< TDSPVector_t >::SetSize(), and clDSPVector< TDSPVector_t >::Size().

5.11.2.77 template<class TDSPVector_t> void clDSPVector< TDSPVector_t >::CartToPolar (clDSPVector< double > & *Magn*, clDSPVector< double > & *Phase*) [inline]

Definition at line 451 of file DSPVector.hh.

References clDSPOp::CartToPolar(), clDSPVector< TDSPVector_t >::DSP, clDSPVector< TDSPVector_t >::Ptr(), clDSPVector< TDSPVector_t >::SetSize(), and clDSPVector< TDSPVector_t >::Size().

5.11.2.78 template<class TDSPVector_t> clDSPVector& clDSPVector< TDSPVector_t >::PolarToCart () [inline]

Definition at line 458 of file DSPVector.hh.

References clDSPVector< TDSPVector_t >::DSP, clDSPOp::PolarToCart(), clDSPVector< TDSPVector_t >::Ptr(), and clDSPVector< TDSPVector_t >::Size().

5.11.2.79 template<class TDSPVector_t> clDSPVector& clDSPVector< TDSPVector_t >::PolarToCart (clDSPVector< TDSPVector_t > & *Polar*) [inline]

Definition at line 463 of file DSPVector.hh.

References clDSPVector< TDSPVector_t >::DSP, clDSPOp::PolarToCart(), clDSPVector< TDSPVector_t >::Ptr(), clDSPVector< TDSPVector_t >::SetSize(), and clDSPVector< TDSPVector_t >::Size().

5.11.2.80 template<class TDSPVector_t> void clDSPVector< TDSPVector_t >::PolarToCart (clDSPVector< float > & *Real*, clDSPVector< float > & *Imag*) [inline]

Definition at line 469 of file DSPVector.hh.

References clDSPVector< TDSPVector_t >::DSP, clDSPOp::PolarToCart(), clDSPVector< TDSPVector_t >::Ptr(), clDSPVector< TDSPVector_t >::SetSize(), and clDSPVector< TDSPVector_t >::Size().

5.11.2.81 template<class TDSPVector_t> void clDSPVector< TDSPVector_t >::PolarToCart (clDSPVector< double > & *Real*, clDSPVector< double > & *Imag*) `[inline]`

Definition at line 476 of file DSPVector.hh.

References clDSPVector< TDSPVector_t >::DSP, clDSPOp::PolarToCart(), clDSPVector< TDSPVector_t >::Ptr(), clDSPVector< TDSPVector_t >::SetSize(), and clDSPVector< TDSPVector_t >::Size().

5.11.2.82 template<class TDSPVector_t> clDSPVector& clDSPVector< TDSPVector_t >::Magnitude (clDSPVector< stSCplx > & *Src*) `[inline]`

Definition at line 483 of file DSPVector.hh.

References clDSPVector< TDSPVector_t >::DSP, DSPV_SHORTER, clDSPOp::Magnitude(), clDSPVector< TDSPVector_t >::Ptr(), clDSPVector< TDSPVector_t >::SetSize(), and clDSPVector< TDSPVector_t >::Size().

5.11.2.83 template<class TDSPVector_t> clDSPVector& clDSPVector< TDSPVector_t >::Magnitude (clDSPVector< stDCplx > & *Src*) `[inline]`

Definition at line 489 of file DSPVector.hh.

References clDSPVector< TDSPVector_t >::DSP, DSPV_SHORTER, clDSPOp::Magnitude(), clDSPVector< TDSPVector_t >::Ptr(), clDSPVector< TDSPVector_t >::SetSize(), and clDSPVector< TDSPVector_t >::Size().

5.11.2.84 template<class TDSPVector_t> clDSPVector& clDSPVector< TDSPVector_t >::Power (clDSPVector< stSCplx > & *Src*) `[inline]`

Definition at line 495 of file DSPVector.hh.

References clDSPVector< TDSPVector_t >::DSP, DSPV_SHORTER, clDSPOp::Power(), clDSPVector< TDSPVector_t >::Ptr(), clDSPVector< TDSPVector_t >::SetSize(), and clDSPVector< TDSPVector_t >::Size().

5.11.2.85 template<class TDSPVector_t> clDSPVector& clDSPVector< TDSPVector_t >::Power (clDSPVector< stDCplx > & *Src*) `[inline]`

Definition at line 501 of file DSPVector.hh.

References clDSPVector< TDSPVector_t >::DSP, DSPV_SHORTER, clDSPOp::Power(), clDSPVector< TDSPVector_t >::Ptr(), clDSPVector< TDSPVector_t >::SetSize(), and clDSPVector< TDSPVector_t >::Size().

5.11.2.86 template<class TDSPVector_t> clDSPVector& clDSPVector< TDSPVector_t >::Phase (clDSPVector< stSCplx > & *Src*) [inline]

Definition at line 507 of file DSPVector.hh.

References clDSPVector< TDSPVector_t >::DSP, DSPV_SHORTER, clDSPOp::Phase(), clDSPVector< TDSPVector_t >::Ptr(), clDSPVector< TDSPVector_t >::SetSize(), and clDSPVector< TDSPVector_t >::Size().

5.11.2.87 template<class TDSPVector_t> clDSPVector& clDSPVector< TDSPVector_t >::Phase (clDSPVector< stDCplx > & *Src*) [inline]

Definition at line 513 of file DSPVector.hh.

References clDSPVector< TDSPVector_t >::DSP, DSPV_SHORTER, clDSPOp::Phase(), clDSPVector< TDSPVector_t >::Ptr(), clDSPVector< TDSPVector_t >::SetSize(), and clDSPVector< TDSPVector_t >::Size().

5.11.2.88 template<class TDSPVector_t> void clDSPVector< TDSPVector_t >::PowerPhase (clDSPVector< float > & *Power*, clDSPVector< float > & *Phase*) [inline]

Definition at line 519 of file DSPVector.hh.

References clDSPVector< TDSPVector_t >::DSP, clDSPOp::PowerPhase(), clDSPVector< TDSPVector_t >::Ptr(), clDSPVector< TDSPVector_t >::SetSize(), and clDSPVector< TDSPVector_t >::Size().

5.11.2.89 template<class TDSPVector_t> void clDSPVector< TDSPVector_t >::PowerPhase (clDSPVector< double > & *Power*, clDSPVector< double > & *Phase*) [inline]

Definition at line 526 of file DSPVector.hh.

References clDSPVector< TDSPVector_t >::DSP, clDSPOp::PowerPhase(), clDSPVector< TDSPVector_t >::Ptr(), clDSPVector< TDSPVector_t >::SetSize(), and clDSPVector< TDSPVector_t >::Size().

5.11.2.90 template<class TDSPVector_t> clDSPVector& clDSPVector< TDSPVector_t >::Decimate (long *lFactor*) [inline]

Definition at line 533 of file DSPVector.hh.

References clDSPOp::Decimate(), clDSPVector< TDSPVector_t >::DSP, clDSPVector< TDSPVector_t >::Ptr(), and clDSPVector< TDSPVector_t >::Size().

5.11.2.91 template<class TDSPVector_t> clDSPVector& clDSPVector< TDSPVector_t >::Decimate (clDSPVector< TDSPVector_t > & *Src*, long *lFactor*) `[inline]`

Definition at line 538 of file DSPVector.hh.

References clDSPOp::Decimate(), clDSPVector< TDSPVector_t >::DSP, clDSPVector< TDSPVector_t >::Ptr(), clDSPVector< TDSPVector_t >::SetSize(), and clDSPVector< TDSPVector_t >::Size().

5.11.2.92 template<class TDSPVector_t> clDSPVector& clDSPVector< TDSPVector_t >::DecimateAvg (clDSPVector< TDSPVector_t > & *Src*, long *lFactor*) `[inline]`

Definition at line 544 of file DSPVector.hh.

References clDSPOp::DecimateAvg(), clDSPVector< TDSPVector_t >::DSP, clDSPVector< TDSPVector_t >::Ptr(), clDSPVector< TDSPVector_t >::SetSize(), and clDSPVector< TDSPVector_t >::Size().

5.11.2.93 template<class TDSPVector_t> clDSPVector& clDSPVector< TDSPVector_t >::Interpolate (clDSPVector< TDSPVector_t > & *Src*, long *lFactor*) `[inline]`

Definition at line 550 of file DSPVector.hh.

References clDSPVector< TDSPVector_t >::DSP, clDSPOp::Interpolate(), clDSPVector< TDSPVector_t >::Ptr(), clDSPVector< TDSPVector_t >::SetSize(), and clDSPVector< TDSPVector_t >::Size().

5.11.2.94 template<class TDSPVector_t> clDSPVector& clDSPVector< TDSPVector_t >::InterpolateAvg (clDSPVector< TDSPVector_t > & *Src*, long *lFactor*) `[inline]`

Definition at line 556 of file DSPVector.hh.

References clDSPVector< TDSPVector_t >::DSP, clDSPOp::InterpolateAvg(), clDSPVector< TDSPVector_t >::Ptr(), clDSPVector< TDSPVector_t >::SetSize(), and clDSPVector< TDSPVector_t >::Size().

5.11.2.95 template<class TDSPVector_t> clDSPVector& clDSPVector< TDSPVector_t >::Resample (clDSPVector< TDSPVector_t > & *Src*) `[inline]`

Definition at line 562 of file DSPVector.hh.

References clDSPVector< TDSPVector_t >::DSP, clDSPVector< TDSPVector_t >::Ptr(), clDSPOp::Resample(), and clDSPVector< TDSPVector_t >::Size().

5.11.2.96 template<class TDSPVector_t> clDSPVector& clDSPVector< TDSPVector_t >::ResampleAvg (clDSPVector< TDSPVector_t > & *Src*) [inline]

Definition at line 567 of file DSPVector.hh.

References clDSPVector< TDSPVector_t >::DSP, clDSPVector< TDSPVector_-t >::Ptr(), clDSPOp::ResampleAvg(), and clDSPVector< TDSPVector_t >::Size().

5.11.2.97 template<class TDSPVector_t> clDSPVector& clDSPVector< TDSPVector_t >::WinBartlett (long *lWinSize*) [inline]

Definition at line 573 of file DSPVector.hh.

References clDSPVector< TDSPVector_t >::DSP, clDSPVector< TDSPVector_-t >::Ptr(), clDSPVector< TDSPVector_t >::SetSize(), clDSPVector< TDSPVector_t >::Size(), and clDSPOp::WinBartlett().

5.11.2.98 template<class TDSPVector_t> clDSPVector& clDSPVector< TDSPVector_t >::WinBlackman (long *lWinSize*) [inline]

Definition at line 579 of file DSPVector.hh.

References clDSPVector< TDSPVector_t >::DSP, clDSPVector< TDSPVector_-t >::Ptr(), clDSPVector< TDSPVector_t >::SetSize(), clDSPVector< TDSPVector_t >::Size(), and clDSPOp::WinBlackman().

5.11.2.99 template<class TDSPVector_t> clDSPVector& clDSPVector< TDSPVector_t >::WinBlackman (long *lWinSize*, TDSPVector_t *fAlpha*) [inline]

Definition at line 585 of file DSPVector.hh.

References clDSPVector< TDSPVector_t >::DSP, clDSPVector< TDSPVector_-t >::Ptr(), clDSPVector< TDSPVector_t >::SetSize(), clDSPVector< TDSPVector_t >::Size(), and clDSPOp::WinBlackman().

5.11.2.100 template<class TDSPVector_t> clDSPVector& clDSPVector< TDSPVector_t >::WinBlackmanHarris (long *lWinSize*) [inline]

Definition at line 591 of file DSPVector.hh.

References clDSPVector< TDSPVector_t >::DSP, clDSPVector< TDSPVector_-t >::Ptr(), clDSPVector< TDSPVector_t >::SetSize(), clDSPVector< TDSPVector_t >::Size(), and clDSPOp::WinBlackmanHarris().

5.11.2.101 template<class TDSPVector_t> clDSPVector& clDSPVector< TDSPVector_t >::WinCosTapered (long *lWinSize*) [inline]

Definition at line 597 of file DSPVector.hh.

References clDSPVector< TDSPVector_t >::DSP, clDSPVector< TDSPVector_t >::Ptr(), clDSPVector< TDSPVector_t >::SetSize(), clDSPVector< TDSPVector_t >::Size(), and clDSPOp::WinCosTapered().

5.11.2.102 template<class TDSPVector_t> clDSPVector& clDSPVector< TDSPVector_t >::WinExactBlackman (long *lWinSize*) [inline]

Definition at line 603 of file DSPVector.hh.

References clDSPVector< TDSPVector_t >::DSP, clDSPVector< TDSPVector_t >::Ptr(), clDSPVector< TDSPVector_t >::SetSize(), clDSPVector< TDSPVector_t >::Size(), and clDSPOp::WinExactBlackman().

5.11.2.103 template<class TDSPVector_t> clDSPVector& clDSPVector< TDSPVector_t >::WinExp (TDSPVector_t *fZ*, long *lWinSize*) [inline]

Definition at line 609 of file DSPVector.hh.

References clDSPVector< TDSPVector_t >::DSP, clDSPVector< TDSPVector_t >::Ptr(), clDSPVector< TDSPVector_t >::SetSize(), clDSPVector< TDSPVector_t >::Size(), and clDSPOp::WinExp().

5.11.2.104 template<class TDSPVector_t> clDSPVector& clDSPVector< TDSPVector_t >::WinFlatTop (long *lWinSize*) [inline]

Definition at line 615 of file DSPVector.hh.

References clDSPVector< TDSPVector_t >::DSP, clDSPVector< TDSPVector_t >::Ptr(), clDSPVector< TDSPVector_t >::SetSize(), clDSPVector< TDSPVector_t >::Size(), and clDSPOp::WinFlatTop().

5.11.2.105 template<class TDSPVector_t> clDSPVector& clDSPVector< TDSPVector_t >::WinGenericCos (clDSPVector< TDSPVector_t > & *Coeffs*, long *lWinSize*) [inline]

Definition at line 621 of file DSPVector.hh.

References clDSPVector< TDSPVector_t >::DSP, clDSPVector< TDSPVector_t >::Ptr(), clDSPVector< TDSPVector_t >::SetSize(), clDSPVector< TDSPVector_t >::Size(), and clDSPOp::WinGenericCos().

5.11.2.106 template<class TDSPVector_t> clDSPVector& clDSPVector< TDSPVector_t >::WinHamming (long *lWinSize*) [inline]

Definition at line 628 of file DSPVector.hh.

References clDSPVector< TDSPVector_t >::DSP, clDSPVector< TDSPVector_-t >::Ptr(), clDSPVector< TDSPVector_t >::SetSize(), clDSPVector< TDSPVector_t >::Size(), and clDSPOp::WinHamming().

5.11.2.107 template<class TDSPVector_t> clDSPVector& clDSPVector< TDSPVector_t >::WinHanning (long *lWinSize*) [inline]

Definition at line 634 of file DSPVector.hh.

References clDSPVector< TDSPVector_t >::DSP, clDSPVector< TDSPVector_-t >::Ptr(), clDSPVector< TDSPVector_t >::SetSize(), clDSPVector< TDSPVector_t >::Size(), and clDSPOp::WinHanning().

5.11.2.108 template<class TDSPVector_t> clDSPVector& clDSPVector< TDSPVector_t >::WinKaiser (TDSPVector_t *fBeta*, long *lWinSize*) [inline]

Definition at line 640 of file DSPVector.hh.

References clDSPVector< TDSPVector_t >::DSP, clDSPVector< TDSPVector_-t >::Ptr(), clDSPVector< TDSPVector_t >::SetSize(), clDSPVector< TDSPVector_t >::Size(), and clDSPOp::WinKaiser().

5.11.2.109 template<class TDSPVector_t> clDSPVector& clDSPVector< TDSPVector_t >::WinKaiserBessel (TDSPVector_t *fAlpha*, long *lWinSize*) [inline]

Definition at line 646 of file DSPVector.hh.

References clDSPVector< TDSPVector_t >::DSP, clDSPVector< TDSPVector_-t >::Ptr(), clDSPVector< TDSPVector_t >::SetSize(), clDSPVector< TDSPVector_t >::Size(), and clDSPOp::WinKaiserBessel().

5.11.2.110 template<class TDSPVector_t> clDSPVector& clDSPVector< TDSPVector_t >::WinTukey (long *lWinSize*) [inline]

Definition at line 652 of file DSPVector.hh.

References clDSPVector< TDSPVector_t >::DSP, clDSPVector< TDSPVector_-t >::Ptr(), clDSPVector< TDSPVector_t >::SetSize(), clDSPVector< TDSPVector_t >::Size(), and clDSPOp::WinTukey().

5.11.2.111 template<class TDSPVector_t> clDSPVector& clDSPVector< TDSPVector_t >::WinDolphChebyshev (TDSPVector_t *fGamma*, long *lWinSize*) [inline]

Definition at line 658 of file DSPVector.hh.

References clDSPVector< TDSPVector_t >::DSP, clDSPVector< TDSPVector_t >::Ptr(), clDSPVector< TDSPVector_t >::SetSize(), clDSPVector< TDSPVector_t >::Size(), and clDSPOp::WinDolphChebyshev().

5.11.2.112 template<class TDSPVector_t> clDSPVector& clDSPVector< TDSPVector_t >::Mix (clDSPVector< TDSPVector_t > & *Src*, long *lChCount*) [inline]

Definition at line 665 of file DSPVector.hh.

References clDSPVector< TDSPVector_t >::DSP, clDSPOp::Mix(), clDSPVector< TDSPVector_t >::Ptr(), clDSPVector< TDSPVector_t >::SetSize(), and clDSPVector< TDSPVector_t >::Size().

5.11.2.113 template<class TDSPVector_t> clDSPVector& clDSPVector< TDSPVector_t >::Extract (clDSPVector< TDSPVector_t > & *Src*, long *lCh*, long *lChCount*) [inline]

Definition at line 671 of file DSPVector.hh.

References clDSPVector< TDSPVector_t >::DSP, clDSPOp::Extract(), clDSPVector< TDSPVector_t >::Ptr(), clDSPVector< TDSPVector_t >::SetSize(), and clDSPVector< TDSPVector_t >::Size().

5.11.2.114 template<class TDSPVector_t> clDSPVector& clDSPVector< TDSPVector_t >::Pack (clDSPVector< TDSPVector_t > & *Src*, long *lCh*, long *lChCount*) [inline]

Definition at line 677 of file DSPVector.hh.

References clDSPVector< TDSPVector_t >::DSP, clDSPOp::Pack(), clDSPVector< TDSPVector_t >::Ptr(), clReBufferT< TDSPVector_t >::Resize(), and clDSPVector< TDSPVector_t >::Size().

5.11.2.115 template<class TDSPVector_t> clDSPVector& clDSPVector< TDSPVector_t >::FFTWConvert (clDSPVector< TDSPVector_t > & *Src*) [inline]

Vector must be sized before calling this!

Definition at line 684 of file DSPVector.hh.

References clDSPVector< TDSPVector_t >::DSP, clDSPOp::FFTWConvert(), clDSPVector< TDSPVector_t >::Ptr(), and clDSPVector< TDSPVector_t >::Size().

5.11.2.116 template<class TDSPVector_t> void clDSPVector< TDSPVector_t >::FIRAllocate (clDSPVector< TDSPVector_t > & *Src*) [inline]

Definition at line 690 of file DSPVector.hh.

References clDSPVector< TDSPVector_t >::DSP, clDSPOp::FIRAllocate(), clDSPVector< TDSPVector_t >::Ptr(), and clDSPVector< TDSPVector_t >::Size().

5.11.2.117 template<class TDSPVector_t> clDSPVector& clDSPVector< TDSPVector_t >::FIRFilter () [inline]

Definition at line 694 of file DSPVector.hh.

References clDSPVector< TDSPVector_t >::DSP, clDSPOp::FIRFilter(), clDSPVector< TDSPVector_t >::Ptr(), and clDSPVector< TDSPVector_t >::Size().

5.11.2.118 template<class TDSPVector_t> clDSPVector& clDSPVector< TDSPVector_t >::FIRFilter (clDSPVector< TDSPVector_t > & *Src*) [inline]

Definition at line 699 of file DSPVector.hh.

References clDSPVector< TDSPVector_t >::DSP, DSPV_SHORTER, clDSPOp::FIRFilter(), clDSPVector< TDSPVector_t >::Ptr(), clDSPVector< TDSPVector_t >::SetSize(), and clDSPVector< TDSPVector_t >::Size().

5.11.2.119 template<class TDSPVector_t> void clDSPVector< TDSPVector_t >::FIRFree () [inline]

Definition at line 705 of file DSPVector.hh.

References clDSPVector< TDSPVector_t >::DSP, and clDSPOp::FIRFree().

5.11.2.120 template<class TDSPVector_t> void clDSPVector< TDSPVector_t >::FFTInitialize (long *lWinSize*, bool *bReal*) [inline]

Definition at line 710 of file DSPVector.hh.

References clDSPVector< TDSPVector_t >::DSP, clDSPOp::FFTInitialize(), and clDSPVector< TDSPVector_t >::lFFTSize.

5.11.2.121 template<class TDSPVector_t> clDSPVector& clDSPVector< TDSPVector_t >::FFT (clDSPVector< float > & *Src*) [inline]

Definition at line 715 of file DSPVector.hh.

References clDSPVector< TDSPVector_t >::DSP, clDSPOp::FFTo(), clDSPVector<

TDSPVector_t >::lFFTSize, clDSPVector< TDSPVector_t >::Ptr(), and clDSPVector< TDSPVector_t >::SetSize().

5.11.2.122 template<class TDSPVector_t> clDSPVector& clDSPVector< TDSPVector_t >::FFT (clDSPVector< double > & *Src*) [inline]

Definition at line 721 of file DSPVector.hh.

References clDSPVector< TDSPVector_t >::DSP, clDSPOp::FFTo(), clDSPVector< TDSPVector_t >::lFFTSize, clDSPVector< TDSPVector_t >::Ptr(), and clDSPVector< TDSPVector_t >::SetSize().

5.11.2.123 template<class TDSPVector_t> clDSPVector& clDSPVector< TDSPVector_t >::FFT (clDSPVector< stSCplx > & *Src*) [inline]

Definition at line 727 of file DSPVector.hh.

References clDSPVector< TDSPVector_t >::DSP, clDSPOp::FFTo(), clDSPVector< TDSPVector_t >::lFFTSize, clDSPVector< TDSPVector_t >::Ptr(), and clDSPVector< TDSPVector_t >::SetSize().

5.11.2.124 template<class TDSPVector_t> clDSPVector& clDSPVector< TDSPVector_t >::FFT (clDSPVector< stDCplx > & *Src*) [inline]

Definition at line 733 of file DSPVector.hh.

References clDSPVector< TDSPVector_t >::DSP, clDSPOp::FFTo(), clDSPVector< TDSPVector_t >::lFFTSize, clDSPVector< TDSPVector_t >::Ptr(), and clDSPVector< TDSPVector_t >::SetSize().

5.11.2.125 template<class TDSPVector_t> clDSPVector& clDSPVector< TDSPVector_t >::IFFT (clDSPVector< stSCplx > & *Src*) [inline]

Definition at line 739 of file DSPVector.hh.

References clDSPVector< TDSPVector_t >::DSP, clDSPOp::IFFTo(), clDSPVector< TDSPVector_t >::lFFTSize, clDSPVector< TDSPVector_t >::Ptr(), and clDSPVector< TDSPVector_t >::SetSize().

5.11.2.126 template<class TDSPVector_t> clDSPVector& clDSPVector< TDSPVector_t >::IFFT (clDSPVector< stDCplx > & *Src*) [inline]

Definition at line 745 of file DSPVector.hh.

References clDSPVector< TDSPVector_t >::DSP, clDSPOp::IFFTo(), clDSPVector< TDSPVector_t >::lFFTSize, clDSPVector< TDSPVector_t >::Ptr(), and clDSPVector< TDSPVector_t >::SetSize().

5.11.2.127 template<class TDSPVector_t> void clDSPVector< TDSPVector_t >::FFTUninitialize () `[inline]`

Definition at line 751 of file DSPVector.hh.

References clDSPVector< TDSPVector_t >::DSP, and clDSPOp::FFTUninitialize().

5.11.2.128 template<class TDSPVector_t> void clDSPVector< TDSPVector_t >::DCTInitialize (long *lWinSize*) `[inline]`

Definition at line 756 of file DSPVector.hh.

References clDSPOp::DCTInitialize(), clDSPVector< TDSPVector_t >::DSP, and clDSPVector< TDSPVector_t >::lFFTSize.

5.11.2.129 template<class TDSPVector_t> void clDSPVector< TDSPVector_t >::DCTUninitialize () `[inline]`

Definition at line 761 of file DSPVector.hh.

References clDSPOp::DCTUninitialize(), and clDSPVector< TDSPVector_t >::DSP.

5.11.2.130 template<class TDSPVector_t> clDSPVector& clDSPVector< TDSPVector_t >::DCT () `[inline]`

Definition at line 765 of file DSPVector.hh.

References clDSPOp::DCTi(), clDSPVector< TDSPVector_t >::DSP, and clDSPVector< TDSPVector_t >::Ptr().

5.11.2.131 template<class TDSPVector_t> clDSPVector& clDSPVector< TDSPVector_t >::DCT (clDSPVector< float > & *Src*) `[inline]`

Definition at line 770 of file DSPVector.hh.

References clDSPOp::DCTo(), clDSPVector< TDSPVector_t >::DSP, clDSPVector< TDSPVector_t >::lFFTSize, clDSPVector< TDSPVector_t >::Ptr(), and clDSPVector< TDSPVector_t >::SetSize().

5.11.2.132 template<class TDSPVector_t> clDSPVector& clDSPVector< TDSPVector_t >::DCT (clDSPVector< double > & *Src*) `[inline]`

Definition at line 776 of file DSPVector.hh.

References clDSPOp::DCTo(), clDSPVector< TDSPVector_t >::DSP, clDSPVector< TDSPVector_t >::lFFTSize, clDSPVector< TDSPVector_t >::Ptr(), and clDSPVector< TDSPVector_t >::SetSize().

5.11.2.133 template<class TDSPVector_t> **clDSPVector**& **clDSPVector**< TDSPVector_t >::IDCT () [inline]

Definition at line 782 of file DSPVector.hh.

References clDSPVector< TDSPVector_t >::DSP, clDSPOp::IDCTi(), and clDSPVector< TDSPVector_t >::Ptr().

5.11.2.134 template<class TDSPVector_t> **clDSPVector**& **clDSPVector**< TDSPVector_t >::IDCT (**clDSPVector**< float > & *Src*) [inline]

Definition at line 787 of file DSPVector.hh.

References clDSPVector< TDSPVector_t >::DSP, clDSPOp::IDCTo(), clDSPVector< TDSPVector_t >::lFFTSize, clDSPVector< TDSPVector_t >::Ptr(), and clDSPVector< TDSPVector_t >::SetSize().

5.11.2.135 template<class TDSPVector_t> **clDSPVector**& **clDSPVector**< TDSPVector_t >::IDCT (**clDSPVector**< double > & *Src*) [inline]

Definition at line 793 of file DSPVector.hh.

References clDSPVector< TDSPVector_t >::DSP, clDSPOp::IDCTo(), clDSPVector< TDSPVector_t >::lFFTSize, clDSPVector< TDSPVector_t >::Ptr(), and clDSPVector< TDSPVector_t >::SetSize().

5.11.3 Member Data Documentation

5.11.3.1 template<class TDSPVector_t> long **clDSPVector**< TDSPVector_t >::**lFFTSize** [private]

Definition at line 37 of file DSPVector.hh.

Referenced by clDSPVector< TDSPVector_t >::clDSPVector(), clDSPVector< TDSPVector_t >::DCT(), clDSPVector< TDSPVector_t >::DCTInitialize(), clDSPVector< TDSPVector_t >::FFT(), clDSPVector< TDSPVector_t >::FFTInitialize(), clDSPVector< TDSPVector_t >::IDCT(), and clDSPVector< TDSPVector_t >::IFFT().

5.11.3.2 template<class TDSPVector_t> clDSPOp clDSPVector< TDSPVector_t >::DSP [private]

Definition at line 38 of file DSPVector.hh.

Referenced by clDSPVector< TDSPVector_t >::Abs(), clDSPVector< TDSPVector_t >::AutoCorr(), clDSPVector< TDSPVector_t >::CartToPolar(), clDSPVector< TDSPVector_t >::Clip(), clDSPVector< TDSPVector_t >::ClipZero(), clDSPVector< TDSPVector_t >::Conv(), clDSPVector< TDSPVector_t >::Convert(), clDSPVector< TDSPVector_t >::Corr(), clDSPVector< TDSPVector_t >::CrossCorr(), clDSPVector< TDSPVector_t >::DCT(), clDSPVector< TDSPVector_t >::DCTInitialize(), clDSPVector< TDSPVector_t >::DCTUninitialize(), clDSPVector< TDSPVector_t >::Decimate(), clDSPVector< TDSPVector_t >::DecimateAvg(), clDSPVector< TDSPVector_t >::Div1x(), clDSPVector< TDSPVector_t >::DotProduct(), clDSPVector< TDSPVector_t >::Energy(), clDSPVector< TDSPVector_t >::Extract(), clDSPVector< TDSPVector_t >::FFT(), clDSPVector< TDSPVector_t >::FFTInitialize(), clDSPVector< TDSPVector_t >::FFTUninitialize(), clDSPVector< TDSPVector_t >::FFTWConvert(), clDSPVector< TDSPVector_t >::FIRAllocate(), clDSPVector< TDSPVector_t >::FIRFilter(), clDSPVector< TDSPVector_t >::FIRFree(), clDSPVector< TDSPVector_t >::IDCT(), clDSPVector< TDSPVector_t >::IFFT(), clDSPVector< TDSPVector_t >::Interpolate(), clDSPVector< TDSPVector_t >::InterpolateAvg(), clDSPVector< TDSPVector_t >::Magnitude(), clDSPVector< TDSPVector_t >::Mean(), clDSPVector< TDSPVector_t >::Median(), clDSPVector< TDSPVector_t >::MinMax(), clDSPVector< TDSPVector_t >::Mix(), clDSPVector< TDSPVector_t >::MulAdd(), clDSPVector< TDSPVector_t >::MulC(), clDSPVector< TDSPVector_t >::Negate(), clDSPVector< TDSPVector_t >::Normalize(), clDSPVector< TDSPVector_t >::operator *(), clDSPVector< TDSPVector_t >::operator *=(), clDSPVector< TDSPVector_t >::operator+(), clDSPVector< TDSPVector_t >::operator+=(), clDSPVector< TDSPVector_t >::operator-(), clDSPVector< TDSPVector_t >::operator-=(), clDSPVector< TDSPVector_t >::operator/(), clDSPVector< TDSPVector_t >::operator/=(), clDSPVector< TDSPVector_t >::Pack(), clDSPVector< TDSPVector_t >::PeakLevel(), clDSPVector< TDSPVector_t >::Phase(), clDSPVector< TDSPVector_t >::PolarToCart(), clDSPVector< TDSPVector_t >::Power(), clDSPVector< TDSPVector_t >::PowerPhase(), clDSPVector< TDSPVector_t >::Resample(), clDSPVector< TDSPVector_t >::ResampleAvg(), clDSPVector< TDSPVector_t >::Reverse(), clDSPVector< TDSPVector_t >::RMS(), clDSPVector< TDSPVector_t >::Scale(), clDSPVector< TDSPVector_t >::Scale01(), clDSPVector< TDSPVector_t >::Set(), clDSPVector< TDSPVector_t >::Sort(), clDSPVector< TDSPVector_t >::Sqrt(), clDSPVector< TDSPVector_t >::Square(), clDSPVector< TDSPVector_t >::StdDev(), clDSPVector< TDSPVector_t >::Sum(), clDSPVector< TDSPVector_t >::Variance(), clDSPVector< TDSPVector_t >::WinBartlett(), clDSPVector< TDSPVector_t >::WinBlackman(), clDSPVector< TDSPVector_t >::WinBlackmanHarris(), clDSPVector< TDSPVector_t >::WinCosTapered(), clDSPVector< TDSPVector_t >::WinDolphChebyshev(), clDSPVector< TDSPVector_t >::WinExactBlackman(), clDSPVector< TDSPVector_t >::WinExp(), clDSPVector< TDSPVector_t >::WinFlatTop(), clDSPVector< TDSPVector_t >::WinGenericCos(), clDSPVector< TDSPVector_t >::WinHamming(), clDSPVector< TDSPVector_t >::WinHanning(), clDSPVector< TDSPVector_t >::WinKaiser(), clDSPVector< TDSPVector_t >::Win-

KaiserBessel(), clDSPVector< TDSPVector_t >::WinTukey(), and clDSPVector< TDSPVector_t >::Zero().

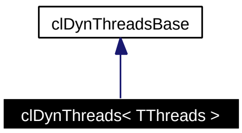
The documentation for this class was generated from the following file:

- DSPVector.hh

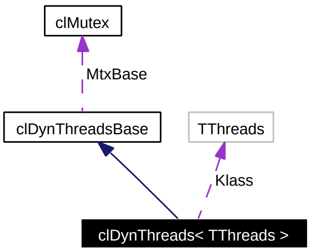
5.12 clDynThreads< TThreads > Class Template Reference

`#include <DynThreads.hh>`

Inheritance diagram for clDynThreads< TThreads >:



Collaboration diagram for clDynThreads< TThreads >:



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 *)

Definition at line 68 of file DynThreads.hh.

Referenced by clDynThreads< TThreads >::Create(), and clDynThreads< TThreads >::InternalCaller().

5.12.1.2 template<class TThreads> typedef struct clDynThreads::_stParams2 clDynThreads< TThreads >::stParams2

Referenced by clDynThreads< TThreads >::Create().

5.12.1.3 template<class TThreads> typedef struct clDynThreads::_stParams2 * clDynThreads< TThreads >::stpParams2

Referenced by clDynThreads< TThreads >::Create(), and clDynThreads< TThreads >::InternalCaller().

5.12.2 Constructor & Destructor Documentation

5.12.2.1 template<class TThreads> clDynThreads< TThreads >::clDynThreads (TThreads & *KlassInst*) `[inline]`

Definition at line 75 of file DynThreads.hh.

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]`

Definition at line 77 of file DynThreads.hh.

References clDynThreadsBase::Create(), clDynThreads< TThreads >::_stParams2::Method, clDynThreads< TThreads >::Method_t, clDynThreads< TThreads >::stParams2, clDynThreads< TThreads >::stpParams2, and clDynThreads< TThreads >::_stParams2::vpParam.

Referenced by main().

5.12.3.2 template<class TThreads> virtual void* clDynThreads< TThreads >::InternalCaller (void * *vpParam*) [inline, virtual]

Implements clDynThreadsBase.

Definition at line 88 of file DynThreads.hh.

References clDynThreads< TThreads >::_stParams2::Method, clDynThreads< TThreads >::Method_t, clDynThreads< TThreads >::stpParams2, and clDynThreads< TThreads >::_stParams2::vpParam.

5.12.4 Member Data Documentation

5.12.4.1 template<class TThreads> TThreads* clDynThreads< TThreads >::Klass

Definition at line 74 of file DynThreads.hh.

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

- DynThreads.hh

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

Definition at line 71 of file DynThreads.hh.

Referenced by clDynThreads< TThreads >::Create(), and clDynThreads< TThreads >::InternalCaller().

5.13.1.2 template<class TThreads> void* clDynThreads< TThreads >::_stParams2::vpParam

Definition at line 72 of file DynThreads.hh.

Referenced by clDynThreads< TThreads >::Create(), and clDynThreads< TThreads >::InternalCaller().

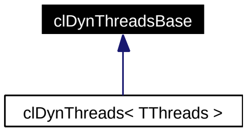
The documentation for this struct was generated from the following file:

- DynThreads.hh

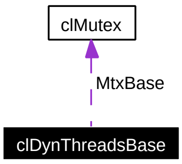
5.14 clDynThreadsBase Class Reference

```
#include <DynThreads.hh>
```

Inheritance diagram for clDynThreadsBase:



Collaboration diagram for clDynThreadsBase:



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

Referenced by Create().

5.14.1.2 typedef struct clDynThreadsBase::_stParams ∗ clDynThreadsBase::stpParams

Referenced by Create(), and WrapDynThreadBase().

5.14.2 Constructor & Destructor Documentation

5.14.2.1 clDynThreadsBase::clDynThreadsBase ()

Definition at line 54 of file DynThreads.cc.

References iThreadCount.

5.14.2.2 clDynThreadsBase::∼clDynThreadsBase () [virtual]

Definition at line 60 of file DynThreads.cc.

References mapThreads, MtxBase, clMutex::Release(), and clMutex::Wait().

5.14.3 Member Function Documentation

5.14.3.1 int clDynThreadsBase::Create (void ∗, bool)

Definition at line 84 of file DynThreads.cc.

References iThreadCount, clDynThreadsBase::_stParams::Klass, mapThreads, MtxBase, clMutex::Release(), stParams, stpParams, clDynThreadsBase::_stParams::vpParam, clMutex::Wait(), and WrapDynThreadBase().

Referenced by clDynThreads< TThreads >::Create().

5.14.3.2 void * clDynThreadsBase::Wait (int)

Definition at line 120 of file DynThreads.cc.

References mapThreads, MtxBase, clMutex::Release(), and clMutex::Wait().

Referenced by main().

5.14.3.3 pthread_t clDynThreadsBase::Self () `[inline]`

Definition at line 57 of file DynThreads.hh.

5.14.3.4 bool clDynThreadsBase::SetSched (pthread_t, int, int)

Definition at line 145 of file DynThreads.cc.

References MtxBase, clMutex::Release(), and clMutex::Wait().

5.14.3.5 virtual void* clDynThreadsBase::InternalCaller (void *) `[pure virtual]`

Implemented in clDynThreads< TThreads >.

Referenced by WrapDynThreadBase().

5.14.4 Member Data Documentation

5.14.4.1 int clDynThreadsBase::iThreadCount `[private]`

Definition at line 44 of file DynThreads.hh.

Referenced by clDynThreadsBase(), and Create().

5.14.4.2 ThreadMap_t clDynThreadsBase::mapThreads `[private]`

Definition at line 45 of file DynThreads.hh.

Referenced by Create(), Wait(), and ~clDynThreadsBase().

5.14.4.3 clMutex clDynThreadsBase::MtxBase `[private]`

Definition at line 46 of file DynThreads.hh.

Referenced by Create(), SetSched(), Wait(), and ~clDynThreadsBase().

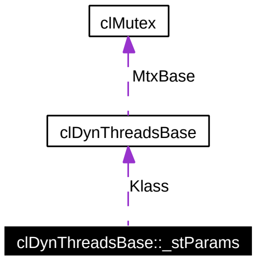
The documentation for this class was generated from the following files:

- DynThreads.hh
- DynThreads.cc

5.15 clDynThreadsBase::_stParams Struct Reference

`#include <DynThreads.hh>`

Collaboration diagram for clDynThreadsBase::_stParams:



Public Attributes

- clDynThreadsBase ∗ Klass
- void ∗ vpParam

5.15.1 Member Data Documentation

5.15.1.1 clDynThreadsBase∗ clDynThreadsBase::_stParams::Klass

Definition at line 50 of file DynThreads.hh.

Referenced by clDynThreadsBase::Create(), and WrapDynThreadBase().

5.15.1.2 void∗ clDynThreadsBase::_stParams::vpParam

Definition at line 51 of file DynThreads.hh.

Referenced by clDynThreadsBase::Create(), and WrapDynThreadBase().

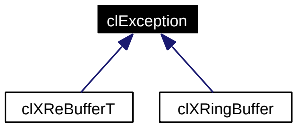
The documentation for this struct was generated from the following file:

- DynThreads.hh

5.16 clException Class Reference

`#include <Exception.hh>`

Inheritance diagram for clException:



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]`

Definition at line 38 of file Exception.hh.

5.16.1.2 clException::clException (int *iErrorCode*) throw () `[inline]`

Definition at line 40 of file Exception.hh.

References iCode, and strMsg.

5.16.1.3 clException::clException (const std::string & *strErrorMsg*) throw () `[inline]`

Definition at line 47 of file Exception.hh.

References iCode, and strMsg.

5.16.1.4 clException::clException (const std::string & *strErrorMsg*, int *iErrorCode*) throw () `[inline]`

Definition at line 52 of file Exception.hh.

References iCode, and strMsg.

5.16.1.5 clException::clException (const char * *cpErrorMsg*) throw () `[inline]`

Definition at line 58 of file Exception.hh.

References iCode, and strMsg.

5.16.1.6 clException::clException (const char * *cpErrorMsg*, int *iErrorCode*) throw () `[inline]`

Definition at line 63 of file Exception.hh.

References iCode, and strMsg.

5.16.1.7 virtual clException::~clException () throw () `[inline, virtual]`

Definition at line 68 of file Exception.hh.

5.16.2 Member Function Documentation

5.16.2.1 clException::operator int () const throw () `[inline]`

Definition at line 70 of file Exception.hh.

References iCode.

5.16.2.2 clException::operator std::string () const throw () `[inline]`

Definition at line 74 of file Exception.hh.

References strMsg.

5.16.2.3 virtual const char* clException::what () const throw () [inline, virtual]

Definition at line 78 of file Exception.hh.

References strMsg.

5.16.3 Member Data Documentation

5.16.3.1 int clException::iCode [private]

Definition at line 35 of file Exception.hh.

Referenced by clException(), and operator int().

5.16.3.2 std::string clException::strMsg [private]

Definition at line 36 of file Exception.hh.

Referenced by clException(), operator std::string(), and what().

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

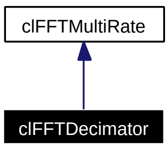
- Exception.hh

5.17 clFFTDecimator Class Reference

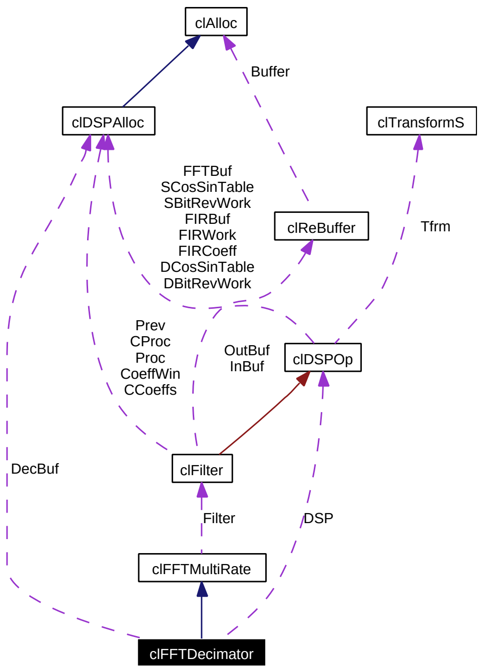
FFT decimation filter class implementation.

`#include <FFTDecimator.hh>`

Inheritance diagram for clFFTDecimator:



Collaboration diagram for clFFTDecimator:



Public Member Functions

- clFFTDecimator ()
- ∼clFFTDecimator ()
- 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

- clDSPAlloc DecBuf
- clDSPOp DSP

5.17.1 Detailed Description

FFT decimation filter class implementation.

Data is filtered using FFT filter and then decimated.

Definition at line 36 of file FFTDecimator.hh.

5.17.2 Constructor & Destructor Documentation

5.17.2.1 clFFTDecimator::clFFTDecimator ()

Definition at line 34 of file FFTDecimator.cc.

5.17.2.2 clFFTDecimator::∼clFFTDecimator ()

Definition at line 39 of file FFTDecimator.cc.

References Uninitialize().

5.17.3 Member Function Documentation

5.17.3.1 void clFFTDecimator::Uninitialize ()

Uninitialize decimator.

Reimplemented from clFFTMultiRate.

Definition at line 45 of file FFTDecimator.cc.

References DecBuf, clAlloc::Free(), and clFFTMultiRate::Uninitialize().

Referenced by clRecDecimator::Uninitialize(), and ∼clFFTDecimator().

5.17.3.2 void clFFTDecimator::Put (const float ∗, long)

Feed data into decimator.

Parameters:

SrcData Source data

SrcCount Source data count

Definition at line 52 of file FFTDecimator.cc.

References clFilter::Put().

Referenced by clRecDecimator::Put().

5.17.3.3 void clFFTDecimator::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.

Definition at line 58 of file FFTDecimator.cc.

References clFilter::Put().

5.17.3.4 bool clFFTDecimator::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?

Definition at line 64 of file FFTDecimator.cc.

References DecBuf, clDSPOp::Decimate(), clFilter::Get(), and clAlloc::Size().

Referenced by clRecDecimator::Get(), and clRecDecimator::Put().

5.17.3.5 bool clFFTDecimator::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.

Definition at line 77 of file FFTDecimator.cc.

References DecBuf, clDSPOp::Decimate(), clFilter::Get(), and clAlloc::Size().

5.17.4 Member Data Documentation

5.17.4.1 clDSPAlloc clFFTDecimator::DecBuf `[private]`

Definition at line 38 of file FFTDecimator.hh.

Referenced by Get(), and Uninitialize().

5.17.4.2 clDSPOp clFFTDecimator::DSP `[private]`

Definition at line 39 of file FFTDecimator.hh.

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

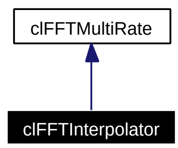
- FFTDecimator.hh
- FFTDecimator.cc

5.18 clFFTInterpolator Class Reference

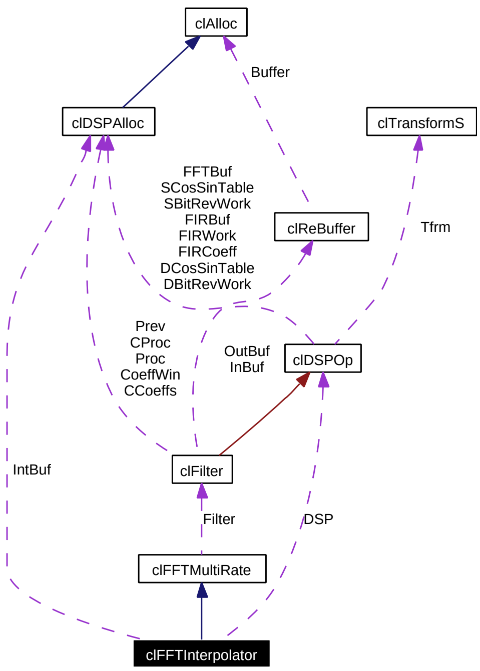
FFT interpolation filter class implementation.

`#include <FFTInterpolator.hh>`

Inheritance diagram for clFFTInterpolator:



Collaboration diagram for clFFTInterpolator:



Public Member Functions

- clFFTInterpolator ()
- ~clFFTInterpolator ()
- 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

- clDSPAlloc IntBuf
- clDSPOp DSP

5.18.1 Detailed Description

FFT interpolation filter class implementation.

Data is interpolated and then filtered using FFT filter.

Definition at line 36 of file FFTInterpolator.hh.

5.18.2 Constructor & Destructor Documentation

5.18.2.1 clFFTInterpolator::clFFTInterpolator ()

Definition at line 34 of file FFTInterpolator.cc.

5.18.2.2 clFFTInterpolator::∼clFFTInterpolator ()

Definition at line 39 of file FFTInterpolator.cc.

References Uninitialize().

5.18.3 Member Function Documentation

5.18.3.1 void clFFTInterpolator::Uninitialize ()

Uninitialize interpolator.

Reimplemented from clFFTMultiRate.

Definition at line 45 of file FFTInterpolator.cc.

References clAlloc::Free(), IntBuf, and clFFTMultiRate::Uninitialize().

Referenced by clRecInterpolator::Uninitialize(), and ∼clFFTInterpolator().

5.18.3.2 void clFFTInterpolator::Put (const float ∗, long)

Feed data into interpolator.

Parameters:

SrcData Source data

SrcCount Source data count

Definition at line 52 of file FFTInterpolator.cc.

References IntBuf, clDSPOp::Interpolate(), clFilter::Put(), and clAlloc::Size().

Referenced by clRecInterpolator::Put().

5.18.3.3 void clFFTInterpolator::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.

Definition at line 64 of file FFTInterpolator.cc.

References IntBuf, clDSPOp::Interpolate(), clFilter::Put(), and clAlloc::Size().

5.18.3.4 bool clFFTInterpolator::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?

Definition at line 76 of file FFTInterpolator.cc.

References clFilter::Get().

Referenced by clRecInterpolator::Get(), and clRecInterpolator::Put().

5.18.3.5 bool clFFTInterpolator::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.

Definition at line 82 of file FFTInterpolator.cc.

References clFilter::Get().

5.18.4 Member Data Documentation

5.18.4.1 clDSPAlloc clFFTInterpolator::IntBuf `[private]`

Definition at line 38 of file FFTInterpolator.hh.

Referenced by Put(), and Uninitialize().

5.18.4.2 clDSPOp clFFTInterpolator::DSP `[private]`

Definition at line 39 of file FFTInterpolator.hh.

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

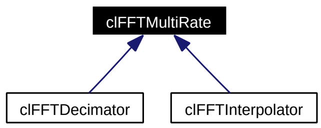
- FFTInterpolator.hh
- FFTInterpolator.cc

5.19 clFFTMultiRate Class Reference

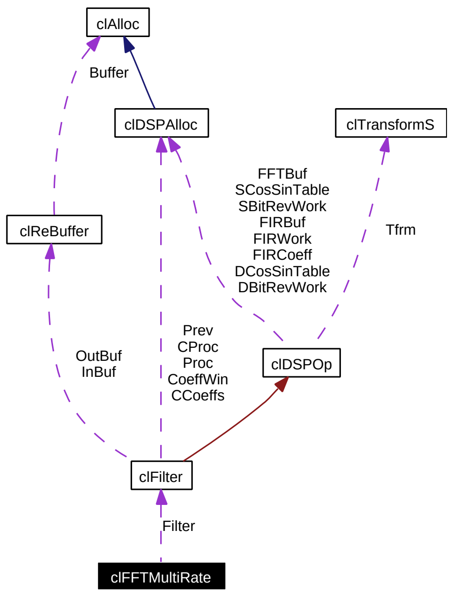
Base class for FFT based multirate filters.

`#include <FFTMultiRate.hh>`

Inheritance diagram for clFFTMultiRate:



Collaboration diagram for clFFTMultiRate:



Public Member Functions

- clFFTMultiRate ()
- ∼clFFTMultiRate ()
- 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.

- clFilter 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.

Definition at line 40 of file FFTMultiRate.hh.

5.19.2 Constructor & Destructor Documentation

5.19.2.1 clFFTMultiRate::clFFTMultiRate ()

Definition at line 34 of file FFTMultiRate.cc.

References bInitialized.

5.19.2.2 clFFTMultiRate::∼clFFTMultiRate ()

Definition at line 40 of file FFTMultiRate.cc.

References bInitialized, and Uninitialize().

5.19.3 Member Function Documentation

5.19.3.1 bool clFFTMultiRate::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?

Definition at line 46 of file FFTMultiRate.cc.

References bInitialized, clFilter::DesignHP(), clFilter::DesignLP(), FFTMULTIRATE_DELTAOMEGA, FFTMULTIRATE_OVERLAP, FFTMULTIRATE_RIPPLERATIO, Filter, clFilter::Initialize(), clFilter::InitializeHP(), clFilter::InitializeLP(), lFactor, lFilterSize, and Uninitialize().

Referenced by clRecInterpolator::Initialize(), and clRecDecimator::Initialize().

5.19.3.2 bool clFFTMultiRate::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.

Definition at line 88 of file FFTMultiRate.cc.

References bInitialized, clFilter::DesignHP(), clFilter::DesignLP(), FFTMULTIRATE_DELTAOMEGA, FFTMULTIRATE_OVERLAP, FFTMULTIRATE_RIPPLERATIO, Filter, clFilter::Initialize(), clFilter::InitializeHP(), clFilter::InitializeLP(), lFactor, lFilterSize, and Uninitialize().

5.19.3.3 void clFFTMultiRate::Uninitialize ()

Uninitialize filter.

Reimplemented in clFFTDecimator, and clFFTInterpolator.

Definition at line 130 of file FFTMultiRate.cc.

References bInitialized, Filter, and clFilter::Uninitialize().

Referenced by Initialize(), clFFTInterpolator::Uninitialize(), clFFTDecimator::Uninitialize(), and ~clFFTMultiRate().

5.19.4 Member Data Documentation

5.19.4.1 bool clFFTMultiRate::bInitialized [protected]

Definition at line 43 of file FFTMultiRate.hh.

Referenced by clFFTMultiRate(), Initialize(), Uninitialize(), and ~clFFTMultiRate().

5.19.4.2 long clFFTMultiRate::lFactor [protected]

Rate change factor.

Definition at line 44 of file FFTMultiRate.hh.

Referenced by Initialize().

5.19.4.3 long clFFTMultiRate::lFilterSize [protected]

Size of filter FFT.

Definition at line 45 of file FFTMultiRate.hh.

Referenced by Initialize().

5.19.4.4 clFilter clFFTMultiRate::Filter [protected]

Definition at line 46 of file FFTMultiRate.hh.

Referenced by Initialize(), and Uninitialize().

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

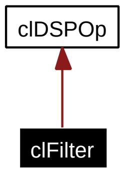
- FFTMultiRate.hh
- FFTMultiRate.cc

5.20 clFilter Class Reference

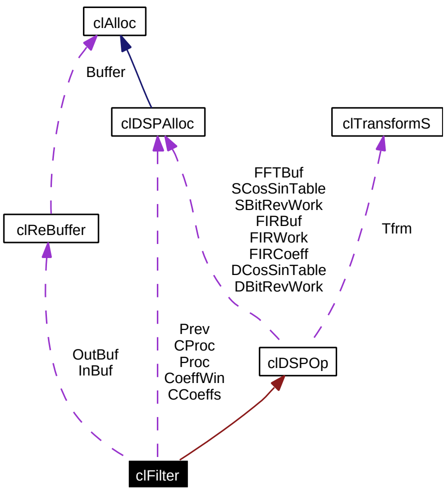
Class implementing FFT-based FIR-filter with design functions.

`#include <Filter.hh>`

Inheritance diagram for clFilter:



Collaboration diagram for clFilter:



Public Member Functions

- clFilter ()
- ~clFilter ()
- bool Initialize (long, const float ∗, float=DSP_FILT_DEF_OVERLAPF, float=DSP_FILT_DEF_BETAF, 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.

Definition at line 47 of file Filter.hh.

5.20.2 Constructor & Destructor Documentation

5.20.2.1 clFilter::clFilter ()

Definition at line 125 of file Filter.cc.

5.20.2.2 clFilter::~clFilter ()

Definition at line 131 of file Filter.cc.

References Uninitialize().

5.20.3 Member Function Documentation

5.20.3.1 void clFilter::InitCoeffsS () [private]

Initialize complex coefficients.

$$\left| \begin{array}{l} \Re_C(i) = 1 \\ \Im_C(i) = 0 \end{array} \right.$$

Definition at line 35 of file Filter.cc.

References CCoeffs, _sSCplx::I, lSpectPoints, _sSCplx::R, ReadyFilterS(), and stpSCplx.

Referenced by Initialize().

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.

Definition at line 49 of file Filter.cc.

References CCoeffs, _sDCplx::I, lSpectPoints, _sDCplx::R, ReadyFilterD(), and stpDCplx.

Referenced by Initialize().

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 $FFT(x)$.

Definition at line 63 of file Filter.cc.

References CCoeffs, CoeffWin, clDSPOp::FFTi(), clDSPOp::IFFTo(), clDSPOp::Mul(), Proc, and stpSCplx.

Referenced by InitCoeffsS(), and SetCoeffs().

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.

Definition at line 76 of file Filter.cc.

References CCoeffs, CoeffWin, clDSPOp::FFTi(), clDSPOp::IFFTo(), clDSPOp::Mul(), Proc, and stpDCplx.

Referenced by InitCoeffsD(), and SetCoeffs().

5.20.3.5 float clFilter::GetKaiserBeta (float *fAlpha*) [private]

Calculate beta value for Kaiser window.

$$\beta = \begin{array}{ll} 0.1102(\alpha_s - 8.7) & ,\alpha_s > 50 \\ 0.5842(\alpha_s - 21)^{0.4} + 0.07886(\alpha_s - 21 & ,21 \leq \alpha_s \leq 50 \\ 0 & ,\alpha_s < 21 \end{array}$$

Parameters:

Alpha Stopband rippleratio in dB

Returns:

Beta for Kaiser window

Definition at line 89 of file Filter.cc.

Referenced by InitializeHP(), and InitializeLP().

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.

Definition at line 107 of file Filter.cc.

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

Definition at line 137 of file Filter.cc.

References CCoeffs, CoeffWin, clDSPOp::Convert(), clDSPOp::Copy(), CProc, clDSPOp::FFTInitialize(), FILTER_SMOOTH_DOLPH_CHEBYSHEV, FILTER_-SMOOTH_KAISER, FILTER_SMOOTH_KAISER_BESSEL, FILTER_SMOOTH_-NONE, InitCoeffsS(), lHalfSize, lNewSize, lOldSize, lSpectPoints, Prev, Proc, cl-DSPOp::Set(), SetCoeffs(), clAlloc::Size(), stSCplx, Uninitialize(), clDSPOp::Win-DolphChebyshev(), clDSPOp::WinKaiser(), clDSPOp::WinKaiserBessel(), and cl-DSPOp::Zero().

Referenced by InitDouble(), InitFloat(), clFFTMultiRate::Initialize(), InitializeHP(), and InitializeLP().

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.

Definition at line 196 of file Filter.cc.

References CCoeffs, CoeffWin, clDSPOp::Copy(), CProc, clDSPOp::FFTInitialize(), FILTER_SMOOTH_DOLPH_CHEBYSHEV, FILTER_SMOOTH_KAISER, FILTER_SMOOTH_KAISER_BESSEL, FILTER_SMOOTH_NONE, InitCoeffsD(), lHalfSize, lNewSize, lOldSize, lSpectPoints, Prev, Proc, clDSPOp::Set(), SetCoeffs(), clAlloc::Size(), stDCplx, Uninitialize(), clDSPOp::WinDolphChebyshev(), clDSPOp::WinKaiser(), clDSPOp::WinKaiserBessel(), and clDSPOp::Zero().

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.

Definition at line 251 of file Filter.cc.

References Initialize().

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.

Definition at line 259 of file Filter.cc.

References Initialize().

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

Definition at line 267 of file Filter.cc.

References DesignLP(), FILTER_SMOOTH_KAISER, GetKaiserBeta(), and Initialize().

Referenced by clFFTMultiRate::Initialize().

5.20.3.12 bool clFilter::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.

Definition at line 295 of file Filter.cc.

References DesignLP(), FILTER_SMOOTH_KAISER, GetKaiserBeta(), and Initialize().

5.20.3.13 bool clFilter::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.

Definition at line 323 of file Filter.cc.

References DesignHP(), FILTER_SMOOTH_KAISER, GetKaiserBeta(), and Initialize().

Referenced by clFFTMultiRate::Initialize().

5.20.3.14 bool clFilter::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.

Definition at line 351 of file Filter.cc.

References DesignHP(), FILTER_SMOOTH_KAISER, GetKaiserBeta(), and Initialize().

5.20.3.15 void clFilter::Uninitialize ()

Uninitialize filter.

Definition at line 379 of file Filter.cc.

References CCoeffs, clReBuffer::Clear(), CoeffWin, CProc, clDSPOp::FFTUninitialize(), clAlloc::Free(), InBuf, OutBuf, Prev, and Proc.

Referenced by Initialize(), clFFTMultiRate::Uninitialize(), and ∼clFilter().

5.20.3.16 void clFilter::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

Definition at line 392 of file Filter.cc.

References CCoeffs, _sSCplx::I, lSpectPoints, _sSCplx::R, ReadyFilterS(), and stpSCplx.

Referenced by DesignBP(), DesignBR(), DesignHP(), DesignLP(), and Initialize().

5.20.3.17 void clFilter::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.

Definition at line 406 of file Filter.cc.

References CCoeffs, _sDCplx::I, lSpectPoints, _sDCplx::R, ReadyFilterD(), and stpDCplx.

5.20.3.18 void clFilter::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.

Definition at line 420 of file Filter.cc.

References CCoeffs, clDSPOp::Copy(), lSpectPoints, ReadyFilterS(), and stpSCplx.

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.

Definition at line 429 of file Filter.cc.

References CCoeffs, clDSPOp::Copy(), lSpectPoints, ReadyFilterD(), and stpDCplx.

5.20.3.20 void clFilter::GetCoeffs (float ∗)

Get coefficients.

Get filter's frequency response.

Parameters:
FiltCoeffs Filter coefficients

Definition at line 438 of file Filter.cc.

References CCoeffs, lSpectPoints, _sSCplx::R, and stpSCplx.

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.

Definition at line 450 of file Filter.cc.

References CCoeffs, lSpectPoints, _sDCplx::R, and stpDCplx.

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.

Definition at line 462 of file Filter.cc.

References CCoeffs, clDSPOp::Copy(), lSpectPoints, and stpSCplx.

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.

Definition at line 470 of file Filter.cc.

References CCoeffs, clDSPOp::Copy(), lSpectPoints, and stpDCplx.

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)

Definition at line 478 of file Filter.cc.

References lOldSize.

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

Definition at line 484 of file Filter.cc.

References CCoeffs, clDSPOp::Copy(), CProc, clDSPOp::FFTi(), clReBuffer::Get(), clDSPOp::IFFTo(), InBuf, lNewSize, lOldSize, lSpectPoints, clDSPOp::Mul(), OutBuf, Prev, Proc, clReBuffer::Put(), and stpSCplx.

Referenced by clFFTInterpolator::Put(), and clFFTDecimator::Put().

5.20.3.26 void clFilter::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.

Definition at line 506 of file Filter.cc.

References CCoeffs, clDSPOp::Copy(), CProc, clDSPOp::FFTi(), clReBuffer::Get(), clDSPOp::IFFTo(), InBuf, lNewSize, lOldSize, lSpectPoints, clDSPOp::Mul(), OutBuf, Prev, Proc, clReBuffer::Put(), and stpDCplx.

5.20.3.27 void clFilter::Put (const float ∗, long, const *stpSCplx*)

Feed data into filter using complex coefficients.

Parameters:

SrcData Source data

SrcCount Source data count

Coeffs Complex coefficients

Definition at line 528 of file Filter.cc.

References clDSPOp::Copy(), CProc, clDSPOp::FFTi(), clReBuffer::Get(), clDSPOp::IFFTo(), InBuf, lNewSize, lOldSize, lSpectPoints, clDSPOp::Mul(), OutBuf, Prev, Proc, clReBuffer::Put(), and stpSCplx.

5.20.3.28 void clFilter::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.

Definition at line 550 of file Filter.cc.

References clDSPOp::Copy(), CProc, clDSPOp::FFTi(), clReBuffer::Get(), clDSPOp::IFFTo(), InBuf, lNewSize, lOldSize, lSpectPoints, clDSPOp::Mul(), OutBuf, Prev, Proc, clReBuffer::Put(), and stpDCplx.

5.20.3.29 bool clFilter::Get (float ∗, long)

Get data from filter.

Parameters:

DestData Destination buffer

DestCount Number of samples

Returns:

Data was available?

Definition at line 572 of file Filter.cc.

References clReBuffer::Get(), and OutBuf.

Referenced by clFFTInterpolator::Get(), and clFFTDecimator::Get().

5.20.3.30 bool clFilter::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.

Definition at line 578 of file Filter.cc.

References clReBuffer::Get(), and OutBuf.

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

Definition at line 584 of file Filter.cc.

References lSpectPoints, clDSPOp::Set(), SetCoeffs(), and clDSPOp::Zero().

Referenced by DesignLP(), clFFTMultiRate::Initialize(), and InitializeLP().

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.

Definition at line 600 of file Filter.cc.

References lSpectPoints, clDSPOp::Set(), SetCoeffs(), and clDSPOp::Zero().

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.

Definition at line 616 of file Filter.cc.

References DesignLP().

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.

Definition at line 628 of file Filter.cc.

References DesignLP().

5.20.3.35 void clFilter::DesignHP (float ∗)

Design high-pass filter.

Definition at line 640 of file Filter.cc.

References lSpectPoints, clDSPOp::Set(), SetCoeffs(), and clDSPOp::Zero().

Referenced by DesignHP(), clFFTMultiRate::Initialize(), and InitializeHP().

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.

Definition at line 655 of file Filter.cc.

References lSpectPoints, clDSPOp::Set(), SetCoeffs(), and clDSPOp::Zero().

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.

Definition at line 670 of file Filter.cc.

References DesignHP().

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.

Definition at line 682 of file Filter.cc.

References DesignHP().

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

Definition at line 694 of file Filter.cc.

References lSpectPoints, clDSPOp::Set(), SetCoeffs(), and clDSPOp::Zero().

Referenced by DesignBP().

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.

Definition at line 712 of file Filter.cc.

References lSpectPoints, clDSPOp::Set(), SetCoeffs(), and clDSPOp::Zero().

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.

Definition at line 730 of file Filter.cc.

References DesignBP().

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.

Definition at line 746 of file Filter.cc.

References DesignBP().

5.20.3.43 void clFilter::DesignBR (float $*$, float $*$)

Design band-reject filter.

Definition at line 762 of file Filter.cc.

References lSpectPoints, clDSPOp::Set(), SetCoeffs(), and clDSPOp::Zero().

Referenced by DesignBR().

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.

Definition at line 780 of file Filter.cc.

References lSpectPoints, clDSPOp::Set(), SetCoeffs(), and clDSPOp::Zero().

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.

Definition at line 798 of file Filter.cc.

References DesignBR().

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.

Definition at line 814 of file Filter.cc.

References DesignBR().

5.20.4 Member Data Documentation

5.20.4.1 bool clFilter::bInitialized `[private]`

Definition at line 49 of file Filter.hh.

5.20.4.2 long clFilter::lFFTSize `[private]`

Definition at line 50 of file Filter.hh.

5.20.4.3 long clFilter::lHalfSize `[private]`

Definition at line 51 of file Filter.hh.

Referenced by Initialize().

5.20.4.4 long clFilter::lOldSize `[private]`

Definition at line 52 of file Filter.hh.

Referenced by GetDelay(), Initialize(), and Put().

5.20.4.5 long clFilter::lNewSize [private]

Definition at line 53 of file Filter.hh.

Referenced by Initialize(), and Put().

5.20.4.6 long clFilter::lSpectPoints [private]

Definition at line 54 of file Filter.hh.

Referenced by DesignBP(), DesignBR(), DesignHP(), DesignLP(), GetCoeffs(), InitCoeffsD(), InitCoeffsS(), Initialize(), Put(), and SetCoeffs().

5.20.4.7 clDSPAlloc clFilter::CoeffWin [private]

Definition at line 55 of file Filter.hh.

Referenced by Initialize(), ReadyFilterD(), ReadyFilterS(), and Uninitialize().

5.20.4.8 clDSPAlloc clFilter::Prev [private]

Definition at line 56 of file Filter.hh.

Referenced by Initialize(), Put(), and Uninitialize().

5.20.4.9 clDSPAlloc clFilter::Proc [private]

Definition at line 57 of file Filter.hh.

Referenced by Initialize(), Put(), ReadyFilterD(), ReadyFilterS(), and Uninitialize().

5.20.4.10 clDSPAlloc clFilter::CCoeffs [private]

Definition at line 58 of file Filter.hh.

Referenced by GetCoeffs(), InitCoeffsD(), InitCoeffsS(), Initialize(), Put(), ReadyFilterD(), ReadyFilterS(), SetCoeffs(), and Uninitialize().

5.20.4.11 clDSPAlloc clFilter::CProc [private]

Definition at line 59 of file Filter.hh.

Referenced by Initialize(), Put(), and Uninitialize().

5.20.4.12 clReBuffer clFilter::InBuf [private]

Definition at line 60 of file Filter.hh.

Referenced by Put(), and Uninitialize().

5.20.4.13 clReBuffer clFilter::OutBuf [private]

Definition at line 61 of file Filter.hh.

Referenced by Get(), Put(), and Uninitialize().

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

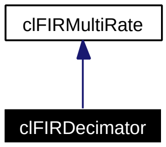
- Filter.hh
- Filter.cc

5.21 clFIRDecimator Class Reference

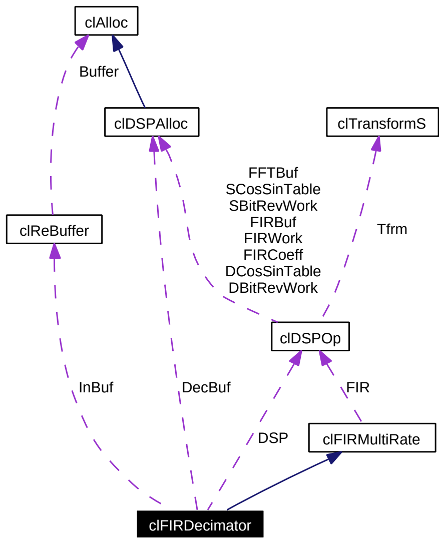
FIR decimation filter class implementation.

`#include <FIRDecimator.hh>`

Inheritance diagram for clFIRDecimator:



Collaboration diagram for clFIRDecimator:



Public Member Functions

- clFIRDecimator ()
- ∼clFIRDecimator ()
- 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

- clDSPAlloc DecBuf
- clDSPOp DSP
- clReBuffer 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.

Definition at line 41 of file FIRDecimator.hh.

5.21.2 Constructor & Destructor Documentation

5.21.2.1 clFIRDecimator::clFIRDecimator ()

Definition at line 34 of file FIRDecimator.cc.

5.21.2.2 clFIRDecimator::∼clFIRDecimator ()

Definition at line 39 of file FIRDecimator.cc.

References Uninitialize().

5.21.3 Member Function Documentation

5.21.3.1 void clFIRDecimator::Uninitialize ()

Uninitialize decimator.

Reimplemented from clFIRMultiRate.

Definition at line 45 of file FIRDecimator.cc.

References clAlloc::Free(), and clFIRMultiRate::Uninitialize().

Referenced by clRecDecimator::Uninitialize(), and ∼clFIRDecimator().

5.21.3.2 void clFIRDecimator::Put (const float ∗, long)

Feed data into decimator.

Parameters:

SrcData Source data

SrcCount Source data count

Definition at line 52 of file FIRDecimator.cc.

References clReBuffer::Put().

Referenced by clRecDecimator::Put().

5.21.3.3 void clFIRDecimator::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.

Definition at line 58 of file FIRDecimator.cc.

References clReBuffer::Put().

5.21.3.4 bool clFIRDecimator::Get (float ∗, long)

Get data from decimator.

Parameters:

DestData Destination buffer

DestCount Number of samples to fetch

Returns:

Success

Definition at line 64 of file FIRDecimator.cc.

References clDSPOp::Decimate(), clDSPOp::FIRFilter(), clReBuffer::Get(), clDSPOp::Mul(), and clAlloc::Size().

Referenced by clRecDecimator::Get(), and clRecDecimator::Put().

5.21.3.5 bool clFIRDecimator::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.

Definition at line 80 of file FIRDecimator.cc.

References clDSPOp::Decimate(), clDSPOp::FIRFilter(), clReBuffer::Get(), clDSPOp::Mul(), and clAlloc::Size().

5.21.4 Member Data Documentation

5.21.4.1 clDSPAlloc clFIRDecimator::DecBuf [private]

Definition at line 43 of file FIRDecimator.hh.

5.21.4.2 clDSPOp clFIRDecimator::DSP [private]

Definition at line 44 of file FIRDecimator.hh.

5.21.4.3 clReBuffer clFIRDecimator::InBuf [private]

Definition at line 45 of file FIRDecimator.hh.

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

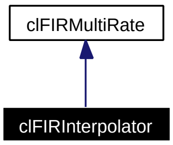
- FIRDecimator.hh
- FIRDecimator.cc

5.22 clFIRInterpolator Class Reference

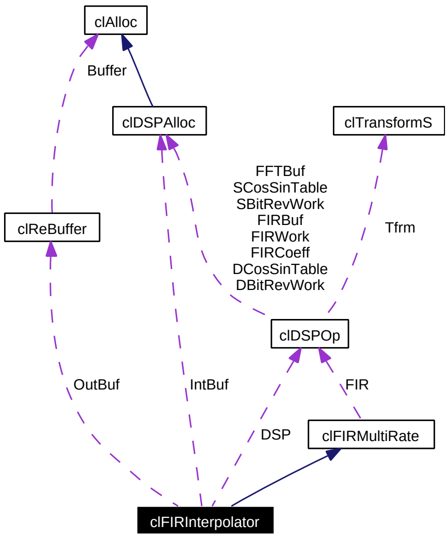
FIR interpolation filter class implementation.

`#include <FIRInterpolator.hh>`

Inheritance diagram for clFIRInterpolator:



Collaboration diagram for clFIRInterpolator:



Public Member Functions

- clFIRInterpolator ()
- ∼clFIRInterpolator ()
- 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

- clDSPAlloc IntBuf
- clDSPOp DSP
- clReBuffer 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.

Definition at line 39 of file FIRInterpolator.hh.

5.22.2 Constructor & Destructor Documentation

5.22.2.1 clFIRInterpolator::clFIRInterpolator ()

Definition at line 34 of file FIRInterpolator.cc.

5.22.2.2 clFIRInterpolator::∼clFIRInterpolator ()

Definition at line 39 of file FIRInterpolator.cc.

References Uninitialize().

5.22.3 Member Function Documentation

5.22.3.1 void clFIRInterpolator::Uninitialize ()

Uninitialize interpolator.

Reimplemented from clFIRMultiRate.

Definition at line 45 of file FIRInterpolator.cc.

References clAlloc::Free(), and clFIRMultiRate::Uninitialize().

Referenced by clRecInterpolator::Uninitialize(), and ∼clFIRInterpolator().

5.22.3.2 void clFIRInterpolator::Put (const float ∗, long)

Feed data into interpolator.

Parameters:

SrcData Source data

SrcCount Source data count

Definition at line 52 of file FIRInterpolator.cc.

References clDSPOp::FIRFilter(), clDSPOp::Interpolate(), clDSPOp::Mul(), clReBuffer::Put(), and clAlloc::Size().

Referenced by clRecInterpolator::Put().

5.22.3.3 void clFIRInterpolator::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.

Definition at line 66 of file FIRInterpolator.cc.

References clDSPOp::FIRFilter(), clDSPOp::Interpolate(), clDSPOp::Mul(), clReBuffer::Put(), and clAlloc::Size().

5.22.3.4 bool clFIRInterpolator::Get (float ∗, long)

Get data from interpolator.

Parameters:

DestData Destination buffer

DestCount Number of samples to fetch

Returns:

Success

Definition at line 80 of file FIRInterpolator.cc.

References clReBuffer::Get().

Referenced by clRecInterpolator::Get(), and clRecInterpolator::Put().

5.22.3.5 bool clFIRInterpolator::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.

Definition at line 86 of file FIRInterpolator.cc.

References clReBuffer::Get().

5.22.4 Member Data Documentation

5.22.4.1 clDSPAlloc clFIRInterpolator::IntBuf `[private]`

Definition at line 41 of file FIRInterpolator.hh.

5.22.4.2 clDSPOp clFIRInterpolator::DSP `[private]`

Definition at line 42 of file FIRInterpolator.hh.

5.22.4.3 clReBuffer clFIRInterpolator::OutBuf `[private]`

Definition at line 43 of file FIRInterpolator.hh.

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

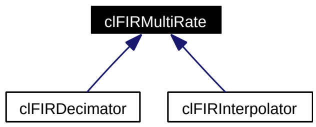
- FIRInterpolator.hh
- FIRInterpolator.cc

5.23 clFIRMultiRate Class Reference

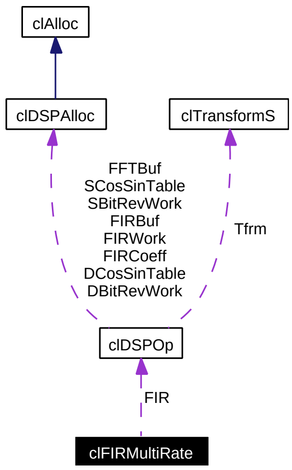
Base class for FIR based multirate filters.

`#include <FIRMultiRate.hh>`

Inheritance diagram for clFIRMultiRate:



Collaboration diagram for clFIRMultiRate:



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.

Definition at line 35 of file FIRMultiRate.hh.

5.23.2 Constructor & Destructor Documentation

5.23.2.1 clFIRMultiRate::clFIRMultiRate ()

Definition at line 51 of file FIRMultiRate.cc.

5.23.2.2 clFIRMultiRate::∼clFIRMultiRate ()

Definition at line 56 of file FIRMultiRate.cc.

References Uninitialize().

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?

Definition at line 62 of file FIRMultiRate.cc.

References fGain, FIR, and clDSPOp::FIRAllocate().

Referenced by clRecInterpolator::Initialize(), and clRecDecimator::Initialize().

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.

Definition at line 123 of file FIRMultiRate.cc.

References dGain, FIR, and clDSPOp::FIRAllocate().

5.23.3.3 void clFIRMultiRate::Uninitialize ()

Uninitialize filter.

Reimplemented in clFIRDecimator, and clFIRInterpolator.

Definition at line 184 of file FIRMultiRate.cc.

References FIR, and clDSPOp::FIRFree().

Referenced by clFIRInterpolator::Uninitialize(), clFIRDecimator::Uninitialize(), and ∼clFIRMultiRate().

5.23.4 Member Data Documentation

5.23.4.1 float clFIRMultiRate::fGain `[protected]`

Filter gain.

Definition at line 38 of file FIRMultiRate.hh.

Referenced by Initialize().

5.23.4.2 double clFIRMultiRate::dGain `[protected]`

Filter gain.

Definition at line 39 of file FIRMultiRate.hh.

Referenced by Initialize().

5.23.4.3 long clFIRMultiRate::lFactor `[protected]`

Rate change factor.

Definition at line 40 of file FIRMultiRate.hh.

5.23.4.4 clDSPOp clFIRMultiRate::FIR `[protected]`

Definition at line 41 of file FIRMultiRate.hh.

Referenced by Initialize(), and Uninitialize().

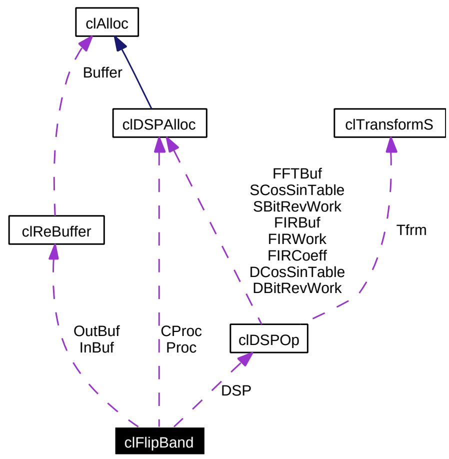
The documentation for this class was generated from the following files:

- FIRMultiRate.hh
- FIRMultiRate.cc

5.24 clFlipBand Class Reference

`#include <FlipBand.hh>`

Collaboration diagram for clFlipBand:



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 ()

Definition at line 26 of file FlipBand.cc.

References lBlockSize.

5.24.1.2 clFlipBand::∼clFlipBand ()

Definition at line 33 of file FlipBand.cc.

References Uninitialize().

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

Definition at line 40 of file FlipBand.cc.

References clDSPOp::FFTInitialize(), lBlockSize, lCBlockSize, clAlloc::Size(), and stSCplx.

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.

Definition at line 51 of file FlipBand.cc.

References clDSPOp::FFTInitialize(), lBlockSize, lCBlockSize, clAlloc::Size(), and stDCplx.

5.24.2.3 void clFlipBand::Uninitialize ()

Uninitialize.

Definition at line 62 of file FlipBand.cc.

References Clear(), clDSPOp::FFTUninitialize(), and clAlloc::Free().

Referenced by ∼clFlipBand().

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

Definition at line 75 of file FlipBand.cc.

References clDSPOp::FFTi(), clReBuffer::Get(), clDSPOp::IFFTo(), lBlockSize, l-CBlockSize, clReBuffer::Put(), clDSPOp::Reverse(), and stpSCplx.

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.

Definition at line 93 of file FlipBand.cc.

References clDSPOp::FFTi(), clReBuffer::Get(), clDSPOp::IFFTo(), lBlockSize, l-CBlockSize, clReBuffer::Put(), clDSPOp::Reverse(), and stpDCplx.

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?

Definition at line 111 of file FlipBand.cc.

References clReBuffer::Get().

5.24.2.7 bool clFlipBand::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.

Definition at line 117 of file FlipBand.cc.

References clReBuffer::Get().

5.24.2.8 void clFlipBand::Clear ()

Clear buffer contents.

Definition at line 123 of file FlipBand.cc.

References clReBuffer::Clear().

Referenced by Uninitialize().

5.24.3 Member Data Documentation

5.24.3.1 bool clFlipBand::bInitialized `[private]`

Definition at line 33 of file FlipBand.hh.

5.24.3.2 long clFlipBand::lBlockSize `[private]`

Definition at line 34 of file FlipBand.hh.

Referenced by clFlipBand(), Initialize(), and Put().

5.24.3.3 long clFlipBand::lCBlockSize `[private]`

Definition at line 35 of file FlipBand.hh.

Referenced by Initialize(), and Put().

5.24.3.4 clDSPAlloc clFlipBand::Proc `[private]`

Definition at line 36 of file FlipBand.hh.

5.24.3.5 clDSPAlloc clFlipBand::CProc [private]

Definition at line 37 of file FlipBand.hh.

5.24.3.6 clDSPOp clFlipBand::DSP [private]

Definition at line 38 of file FlipBand.hh.

5.24.3.7 clReBuffer clFlipBand::InBuf [private]

Definition at line 39 of file FlipBand.hh.

5.24.3.8 clReBuffer clFlipBand::OutBuf [private]

Definition at line 40 of file FlipBand.hh.

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

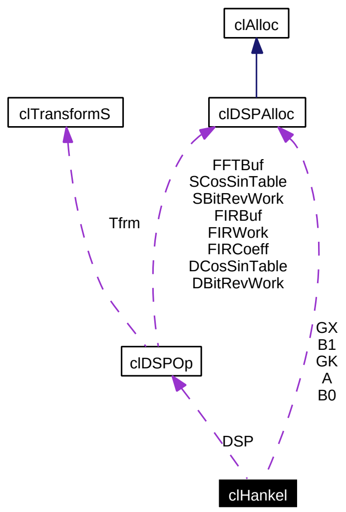
- FlipBand.hh
- FlipBand.cc

5.25 clHankel Class Reference

Class implementation of (modified) Hankel-transform.

`#include <Hankel.hh>`

Collaboration diagram for clHankel:



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 ()

  Unintialize 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!)

Definition at line 58 of file Hankel.hh.

5.25.2 Constructor & Destructor Documentation

5.25.2.1 clHankel::clHankel ()

Definition at line 229 of file Hankel.cc.

5.25.2.2 clHankel::~clHankel ()

Definition at line 235 of file Hankel.cc.

References Uninitialize().

5.25.3 Member Function Documentation

5.25.3.1 void clHankel::InitAbel (const float *) `[private]`

Initialize Abel transformer.

Parameters:

NullPtr NULL pointer

Definition at line 89 of file Hankel.cc.

References A, ABEL_NSE, B0, B1, fpH, fpLambda, and clAlloc::Size().

Referenced by Initialize().

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.

Definition at line 118 of file Hankel.cc.

References A, ABEL_NSE, B0, B1, fpH, fpLambda, and clAlloc::Size().

5.25.3.3 void clHankel::DoAbel (float *, const float *) `[private]`

Do Abel transform.

Parameters:

Dest Destination

Src Source

Definition at line 147 of file Hankel.cc.

References A, ABEL_NSE, B0, B1, and clAlloc::GetPtr().

Referenced by Process0(), and Process1().

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.

Definition at line 184 of file Hankel.cc.

References A, ABEL_NSE, B0, B1, and clAlloc::GetPtr().

5.25.3.5 void clHankel::UninitAbel () `[private]`

Unintialize Abel transformer.

Definition at line 221 of file Hankel.cc.

References A, B0, B1, and clAlloc::Free().

Referenced by Uninitialize().

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

Definition at line 241 of file Hankel.cc.

References clDSPOp::FFTInitialize(), fOutScale0, fOutScale1, GK, GX, InitAbel(), clAlloc::Size(), and stSCplx.

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.

Definition at line 255 of file Hankel.cc.

References dOutScale0, dOutScale1, clDSPOp::FFTInitialize(), GK, GX, InitAbel(), clAlloc::Size(), and stDCplx.

5.25.3.8 void clHankel::Uninitialize ()

Uninitialize, destructor also does this.

Definition at line 269 of file Hankel.cc.

References clAlloc::Free(), GK, GX, and UninitAbel().

Referenced by ∼clHankel().

5.25.3.9 void clHankel::Process0 (float ∗, const float ∗)

Process data (0th order).

Definition:

$$h_0(k) = \int_0^\infty r j_0(kr) f(r) \mathrm{d}r$$

Note:

Input and output is window size / 2 + 1!

Parameters:

Dest Destination

Src Source

Definition at line 278 of file Hankel.cc.

References DoAbel(), GK, GX, _sSCplx::I, clDSPOp::IFFTo(), _sSCplx::R, stpSCplx, and stSCplx.

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.

Definition at line 319 of file Hankel.cc.

References DoAbel(), GK, GX, _sDCplx::I, clDSPOp::IFFTo(), _sDCplx::R, stDCplx, and stpDCplx.

5.25.3.11 void clHankel::Process1 (float ∗, const float ∗)

Process data (1st order).

Definition:

$$h_1(k) = \int_0^\infty r j_1(kr) f(r) \mathrm{d}r$$

Note:

Input and output is window size / 2 + 1!

Parameters:

Dest Destination

Src Source

Definition at line 360 of file Hankel.cc.

References DoAbel(), GK, GX, _sSCplx::I, clDSPOp::IFFTo(), _sSCplx::R, stpSCplx, and stSCplx.

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.

Definition at line 401 of file Hankel.cc.

References DoAbel(), GK, GX, _sDCplx::I, clDSPOp::IFFTo(), _sDCplx::R, stDCplx, and stpDCplx.

5.25.4 Member Data Documentation

5.25.4.1 long clHankel::lSize `[private]`

Definition at line 60 of file Hankel.hh.

5.25.4.2 long clHankel::lFFTSize `[private]`

Definition at line 61 of file Hankel.hh.

5.25.4.3 float clHankel::fOutScale0 `[private]`

Definition at line 62 of file Hankel.hh.

Referenced by Initialize().

5.25.4.4 float clHankel::fOutScale1 `[private]`

Definition at line 63 of file Hankel.hh.

Referenced by Initialize().

5.25.4.5 double clHankel::dOutScale0 `[private]`

Definition at line 64 of file Hankel.hh.

Referenced by Initialize().

5.25.4.6 double clHankel::dOutScale1 [private]

Definition at line 65 of file Hankel.hh.

Referenced by Initialize().

5.25.4.7 clDSPAlloc clHankel::A [private]

Definition at line 66 of file Hankel.hh.

Referenced by DoAbel(), InitAbel(), and UninitAbel().

5.25.4.8 clDSPAlloc clHankel::B0 [private]

Definition at line 67 of file Hankel.hh.

Referenced by DoAbel(), InitAbel(), and UninitAbel().

5.25.4.9 clDSPAlloc clHankel::B1 [private]

Definition at line 68 of file Hankel.hh.

Referenced by DoAbel(), InitAbel(), and UninitAbel().

5.25.4.10 clDSPAlloc clHankel::GX [private]

Definition at line 69 of file Hankel.hh.

Referenced by Initialize(), Process0(), Process1(), and Uninitialize().

5.25.4.11 clDSPAlloc clHankel::GK [private]

Definition at line 70 of file Hankel.hh.

Referenced by Initialize(), Process0(), Process1(), and Uninitialize().

5.25.4.12 clDSPOp clHankel::DSP [private]

Definition at line 71 of file Hankel.hh.

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

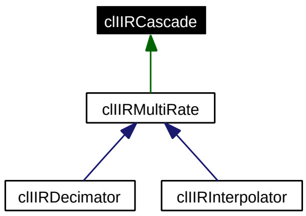
- Hankel.hh
- Hankel.cc

5.26 clIIRCascade Class Reference

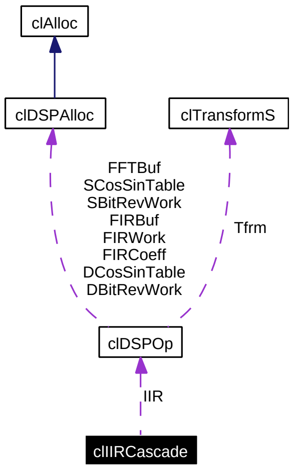
Class to handle cascaded IIR stages as one filter.

`#include <IIRCascade.hh>`

Inheritance diagram for clIIRCascade:



Collaboration diagram for clIIRCascade:



Public Member Functions

- clIIRCascade ()
- ~clIIRCascade ()
- 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
- clDSPOp $*$ IIR

Private Attributes

- bool bInitialized

5.26.1 Detailed Description

Class to handle cascaded IIR stages as one filter.

Definition at line 33 of file IIRCascade.hh.

5.26.2 Constructor & Destructor Documentation

5.26.2.1 clIIRCascade::clIIRCascade ()

Definition at line 26 of file IIRCascade.cc.

References IIR, and lStages.

5.26.2.2 clIIRCascade::~clIIRCascade ()

Definition at line 34 of file IIRCascade.cc.

References Uninitialize().

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

Definition at line 41 of file IIRCascade.cc.

References IIR, clDSPOp::IIRInitialize(), lStages, and Uninitialize().

Referenced by clIIRMultiRate::Initialize().

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.

Definition at line 56 of file IIRCascade.cc.

References IIR, clDSPOp::IIRInitialize(), lStages, and Uninitialize().

5.26.3.3 void clIIRCascade::Uninitialize ()

Uninitialize filter.

Reimplemented in clIIRDecimator, clIIRInterpolator, and clIIRMultiRate.

Definition at line 71 of file IIRCascade.cc.

References IIR, and lStages.

Referenced by Initialize(), clIIRMultiRate::Uninitialize(), and ∼clIIRCascade().

5.26.3.4 void clIIRCascade::Process (float ∗, long)

Filter data in-place.

Parameters:
Vect Source & destination vector
Count Vector length

Definition at line 83 of file IIRCascade.cc.

References IIR, clDSPOp::IIRFilter(), and lStages.

Referenced by clIIRDecimator::Get(), and clIIRInterpolator::Put().

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.

Definition at line 92 of file IIRCascade.cc.

References IIR, clDSPOp::IIRFilter(), and lStages.

5.26.3.6 void clIIRCascade::Process (float *, const float *, long)

Filter data out-of-place.

Parameters:

Dest Destination vector

Src Source vector

Count Vector length

Definition at line 101 of file IIRCascade.cc.

References IIR, clDSPOp::IIRFilter(), and lStages.

5.26.3.7 void clIIRCascade::Process (double *, const double *, long)

Definition at line 111 of file IIRCascade.cc.

References IIR, clDSPOp::IIRFilter(), and lStages.

5.26.3.8 void clIIRCascade::Clear ()

Clear filter feedback chain.

Definition at line 121 of file IIRCascade.cc.

References IIR, clDSPOp::IIRClear(), and lStages.

5.26.4 Member Data Documentation

5.26.4.1 bool clIIRCascade::bInitialized `[private]`

Definition at line 35 of file IIRCascade.hh.

5.26.4.2 long clIIRCascade::lStages [protected]

Definition at line 37 of file IIRCascade.hh.

Referenced by Clear(), clIIRCascade(), Initialize(), Process(), and Uninitialize().

5.26.4.3 clDSPOp* clIIRCascade::IIR [protected]

Definition at line 38 of file IIRCascade.hh.

Referenced by Clear(), clIIRCascade(), Initialize(), Process(), and Uninitialize().

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

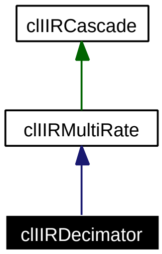
- IIRCascade.hh
- IIRCascade.cc

5.27 clIIRDecimator Class Reference

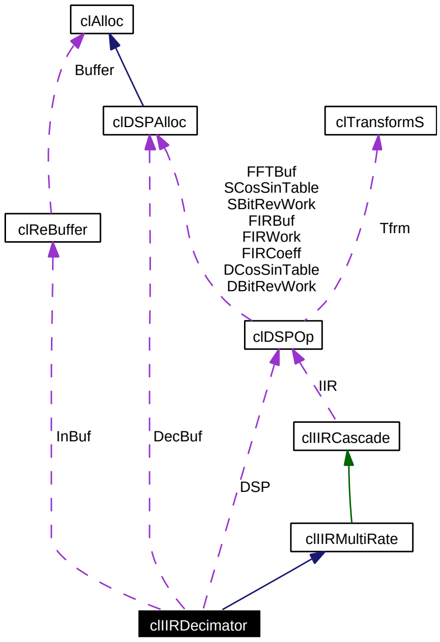
IIR decimation filter class implementation.

`#include <IIRDecimator.hh>`

Inheritance diagram for clIIRDecimator:



Collaboration diagram for clIIRDecimator:



Public Member Functions

- clIIRDecimator ()

- ~clIIRDecimator ()
- 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

- clDSPAlloc DecBuf
- clDSPOp DSP
- clReBuffer 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.

Definition at line 39 of file IIRDecimator.hh.

5.27.2 Constructor & Destructor Documentation

5.27.2.1 clIIRDecimator::clIIRDecimator ()

Definition at line 34 of file IIRDecimator.cc.

5.27.2.2 clIIRDecimator::~clIIRDecimator ()

Definition at line 39 of file IIRDecimator.cc.

References Uninitialize().

5.27.3 Member Function Documentation

5.27.3.1 void clIIRDecimator::Uninitialize ()

Uninitialize decimator.

Reimplemented from clIIRMultiRate.

Definition at line 45 of file IIRDecimator.cc.

References clAlloc::Free(), and clIIRMultiRate::Uninitialize().

Referenced by clRecDecimator::Uninitialize(), and ∼clIIRDecimator().

5.27.3.2 void clIIRDecimator::Put (const float ∗, long)

Feed data into decimator.

Parameters:

SrcData Source data

SrcCount Source data count

Definition at line 52 of file IIRDecimator.cc.

References clReBuffer::Put().

Referenced by clRecDecimator::Put().

5.27.3.3 void clIIRDecimator::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.

Definition at line 58 of file IIRDecimator.cc.

References clReBuffer::Put().

5.27.3.4 bool clIIRDecimator::Get (float ∗, long)

Get data from decimator.

Parameters:

DestData Destination buffer

DestCount Number of samples to fetch

Returns:

Success

Definition at line 64 of file IIRDecimator.cc.

References clDSPOp::Decimate(), clReBuffer::Get(), clIIRCascade::Process(), and clAlloc::Size().

Referenced by clRecDecimator::Get(), and clRecDecimator::Put().

5.27.3.5 bool clIIRDecimator::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.

Definition at line 79 of file IIRDecimator.cc.

References clDSPOp::Decimate(), clReBuffer::Get(), clIIRCascade::Process(), and clAlloc::Size().

5.27.4 Member Data Documentation

5.27.4.1 clDSPAlloc clIIRDecimator::DecBuf `[private]`

Definition at line 41 of file IIRDecimator.hh.

5.27.4.2 clDSPOp clIIRDecimator::DSP `[private]`

Definition at line 42 of file IIRDecimator.hh.

5.27.4.3 clReBuffer clIIRDecimator::InBuf `[private]`

Definition at line 43 of file IIRDecimator.hh.

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

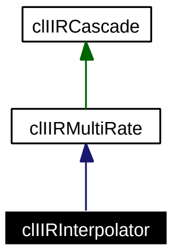
- IIRDecimator.hh
- IIRDecimator.cc

5.28 clIIRInterpolator Class Reference

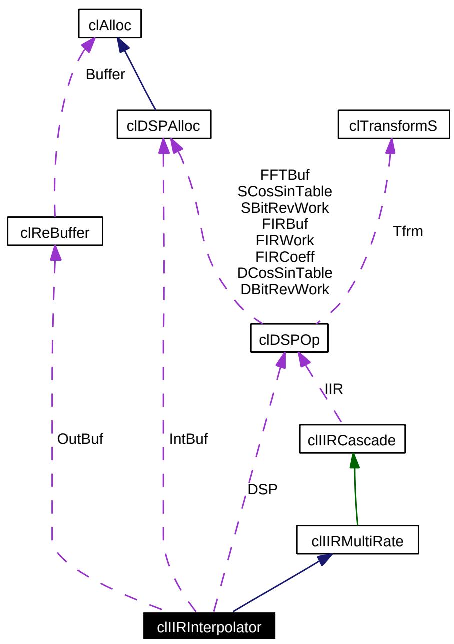
IIR interpolation filter class implementation.

`#include <IIRInterpolator.hh>`

Inheritance diagram for clIIRInterpolator:



Collaboration diagram for clIIRInterpolator:



Public Member Functions

- clIIRInterpolator ()

- ~clIIRInterpolator ()
- 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

- clDSPAlloc IntBuf
- clDSPOp DSP
- clReBuffer 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.

Definition at line 39 of file IIRInterpolator.hh.

5.28.2 Constructor & Destructor Documentation

5.28.2.1 clIIRInterpolator::clIIRInterpolator ()

Definition at line 34 of file IIRInterpolator.cc.

5.28.2.2 clIIRInterpolator::~clIIRInterpolator ()

Definition at line 39 of file IIRInterpolator.cc.

References Uninitialize().

5.28.3 Member Function Documentation

5.28.3.1 void clIIRInterpolator::Uninitialize ()

Uninitialize interpolator.

Reimplemented from clIIRMultiRate.

Definition at line 45 of file IIRInterpolator.cc.

References clAlloc::Free(), and clIIRMultiRate::Uninitialize().

Referenced by clRecInterpolator::Uninitialize(), and ~clIIRInterpolator().

5.28.3.2 void clIIRInterpolator::Put (const float *, long)

Feed data into interpolator.

Parameters:

SrcData Source data

SrcCount Source data count

Definition at line 52 of file IIRInterpolator.cc.

References clDSPOp::Interpolate(), clIIRCascade::Process(), clReBuffer::Put(), and clAlloc::Size().

Referenced by clRecInterpolator::Put().

5.28.3.3 void clIIRInterpolator::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.

Definition at line 65 of file IIRInterpolator.cc.

References clDSPOp::Interpolate(), clIIRCascade::Process(), clReBuffer::Put(), and clAlloc::Size().

5.28.3.4 bool clIIRInterpolator::Get (float *, long)

Get data from interpolator.

Parameters:

DestData Destination buffer

DestCount Number of samples to fetch

Returns:

Success

Definition at line 78 of file IIRInterpolator.cc.

References clReBuffer::Get().

Referenced by clRecInterpolator::Get(), and clRecInterpolator::Put().

5.28.3.5 bool clIIRInterpolator::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.

Definition at line 84 of file IIRInterpolator.cc.

References clReBuffer::Get().

5.28.4 Member Data Documentation

5.28.4.1 clDSPAlloc clIIRInterpolator::IntBuf `[private]`

Definition at line 41 of file IIRInterpolator.hh.

5.28.4.2 clDSPOp clIIRInterpolator::DSP `[private]`

Definition at line 42 of file IIRInterpolator.hh.

5.28.4.3 clReBuffer clIIRInterpolator::OutBuf `[private]`

Definition at line 43 of file IIRInterpolator.hh.

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

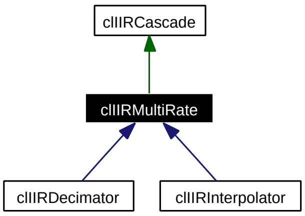
- IIRInterpolator.hh
- IIRInterpolator.cc

5.29 clIIRMultiRate Class Reference

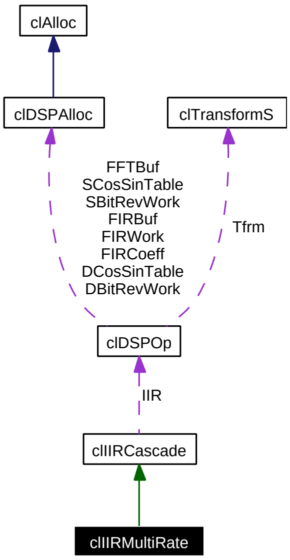
Base class for IIR based multirate filters.

`#include <IIRMultiRate.hh>`

Inheritance diagram for clIIRMultiRate:



Collaboration diagram for clIIRMultiRate:



Public Member Functions

- clIIRMultiRate ()
- ∼clIIRMultiRate ()
- 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.

Definition at line 36 of file IIRMultiRate.hh.

5.29.2 Constructor & Destructor Documentation

5.29.2.1 clIIRMultiRate::clIIRMultiRate ()

Definition at line 47 of file IIRMultiRate.cc.

5.29.2.2 clIIRMultiRate::∼clIIRMultiRate ()

Definition at line 53 of file IIRMultiRate.cc.

References Uninitialize().

5.29.3 Member Function Documentation

5.29.3.1 bool clIIRMultiRate::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?

Definition at line 59 of file IIRMultiRate.cc.

References clIIRCascade::Initialize().

Referenced by clRecInterpolator::Initialize(), and clRecDecimator::Initialize().

5.29.3.2 bool clIIRMultiRate::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.

Definition at line 84 of file IIRMultiRate.cc.

References clIIRCascade::Initialize().

5.29.3.3 void clIIRMultiRate::Uninitialize ()

Uninitialize filter.

Reimplemented from clIIRCascade.

Reimplemented in clIIRDecimator, and clIIRInterpolator.

Definition at line 109 of file IIRMultiRate.cc.

References clIIRCascade::Uninitialize().

Referenced by clIIRInterpolator::Uninitialize(), clIIRDecimator::Uninitialize(), and ~clIIRMultiRate().

5.29.4 Member Data Documentation

5.29.4.1 long clIIRMultiRate::lFactor `[protected]`

Rate change factor.

Definition at line 39 of file IIRMultiRate.hh.

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

- IIRMultiRate.hh
- IIRMultiRate.cc

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 ()

  Desctructor; 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.

Definition at line 37 of file Mutex.hh.

5.30.2 Constructor & Destructor Documentation

5.30.2.1 clMutex::clMutex () [inline]

Constructor; creates and initializes the mutex.

Definition at line 44 of file Mutex.hh.

References pthmMutex.

5.30.2.2 clMutex::~clMutex () [inline]

Desctructor; destroys the mutex.

Definition at line 48 of file Mutex.hh.

References pthmMutex.

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

Definition at line 55 of file Mutex.hh.

References pthmMutex.

Referenced by clDynThreadsBase::Create(), clDynThreadsBase::SetSched(), clDynThreadsBase::Wait(), and clDynThreadsBase::~clDynThreadsBase().

5.30.3.2 bool clMutex::Release () [inline]

Unlock mutex.

Returns:
Success

Definition at line 66 of file Mutex.hh.

References pthmMutex.

Referenced by clDynThreadsBase::Create(), clDynThreadsBase::SetSched(), clDynThreadsBase::Wait(), and clDynThreadsBase::~clDynThreadsBase().

5.30.3.3 bool clMutex::TryLock () [inline]

Try locking mutex without blocking the calling process.

Returns:
Success

Definition at line 77 of file Mutex.hh.

References pthmMutex.

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

Definition at line 89 of file Mutex.hh.

References pthmMutex.

5.30.4 Member Data Documentation

5.30.4.1 pthread_mutex_t clMutex::pthmMutex [private]

Definition at line 39 of file Mutex.hh.

Referenced by clMutex(), GetPtr(), Release(), TryLock(), Wait(), and ~clMutex().

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

- Mutex.hh

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]`

Definition at line 33 of file PthCond.hh.

5.31.1.2 clPthCond::~clPthCond () `[inline]`

Definition at line 34 of file PthCond.hh.

5.31.2 Member Function Documentation

5.31.2.1 void clPthCond::Wait (pth_mutex_t * *pthmCond*) `[inline]`

Definition at line 35 of file PthCond.hh.

5.31.2.2 void clPthCond::Wait (pth_mutex_t * *pthmCond*, pth_event_t *ptheCond*) `[inline]`

Definition at line 37 of file PthCond.hh.

5.31.2.3 void clPthCond::Notify (int *iNotifyAll*) [inline]

Definition at line 39 of file PthCond.hh.

5.31.3 Member Data Documentation

5.31.3.1 pth_cond_t clPthCond::pthcCond [private]

Definition at line 31 of file PthCond.hh.

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

- PthCond.hh

5.32 clPthMutex Class Reference

`#include <PthMutex.hh>`

Public Member Functions

- clPthMutex ()
- ~clPthMutex ()
- void Wait ()
- void Release ()
- pth_mutex_t * GetPtr ()

Private Attributes

- pth_mutex_t pthmMutex

5.32.1 Constructor & Destructor Documentation

5.32.1.1 clPthMutex::clPthMutex () `[inline]`

Definition at line 33 of file PthMutex.hh.

5.32.1.2 clPthMutex::~clPthMutex () `[inline]`

Definition at line 34 of file PthMutex.hh.

5.32.2 Member Function Documentation

5.32.2.1 void clPthMutex::Wait () `[inline]`

Definition at line 35 of file PthMutex.hh.

5.32.2.2 void clPthMutex::Release () `[inline]`

Definition at line 36 of file PthMutex.hh.

5.32.2.3 pth_mutex_t* clPthMutex::GetPtr () [inline]

Definition at line 37 of file PthMutex.hh.

5.32.3 Member Data Documentation

5.32.3.1 pth_mutex_t clPthMutex::pthmMutex [private]

Definition at line 31 of file PthMutex.hh.

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

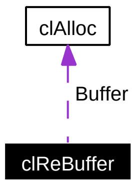
- PthMutex.hh

5.33 clReBuffer Class Reference

Class for splitting data into buffers of different sizes.

`#include <ReBuffer.hh>`

Collaboration diagram for clReBuffer:



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.

Definition at line 36 of file ReBuffer.hh.

5.33.2 Constructor & Destructor Documentation

5.33.2.1 clReBuffer::clReBuffer ()

Definition at line 116 of file ReBuffer.cc.

References lCount, lGetIndex, and lPutIndex.

5.33.2.2 clReBuffer::clReBuffer (const clReBuffer & *CopySrc*) `[inline]`

Definition at line 49 of file ReBuffer.hh.

5.33.2.3 clReBuffer::~clReBuffer ()

Definition at line 125 of file ReBuffer.cc.

5.33.3 Member Function Documentation

5.33.3.1 void clReBuffer::CheckSize (long, long) `[inline, private]`

Definition at line 37 of file ReBuffer.cc.

References Buffer, clAlloc::GetSize(), lCount, lGetIndex, lPutIndex, and clAlloc::Resize().

Referenced by Put().

5.33.3.2 void * clReBuffer::Index (const std::type_info &, long) `[protected]`

Definition at line 69 of file ReBuffer.cc.

References Buffer, and lGetIndex.

5.33.3.3 void * clReBuffer::GetPtr (const std::type_info &) `[protected]`

Definition at line 90 of file ReBuffer.cc.

References Buffer, Clear(), Get(), lCount, lGetIndex, Put(), and clAlloc::Size().

5.33.3.4 void clReBuffer::Put (const float *, long)

Put data into FIFO.

Parameters:

fpSrcData Source buffer

lSrcCount Number of samples in source.

Definition at line 130 of file ReBuffer.cc.

References Buffer, CheckSize(), lCount, and lPutIndex.

Referenced by GetPtr(), clIIRInterpolator::Put(), clIIRDecimator::Put(), clFlipBand::Put(), clFIRInterpolator::Put(), clFIRDecimator::Put(), and clFilter::Put().

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.

Definition at line 152 of file ReBuffer.cc.

References Buffer, CheckSize(), lCount, and lPutIndex.

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

Definition at line 174 of file ReBuffer.cc.

References Buffer, lCount, and lGetIndex.

Referenced by clIIRInterpolator::Get(), clIIRDecimator::Get(), clFlipBand::Get(), clFIRInterpolator::Get(), clFIRDecimator::Get(), clFilter::Get(), GetPtr(), clFlipBand::Put(), and clFilter::Put().

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.

Definition at line 197 of file ReBuffer.cc.

References Buffer, lCount, and lGetIndex.

5.33.3.8 long clReBuffer::GetCount () const `[inline]`

Get number of samples in FIFO.

Returns:

Number of samples

Definition at line 80 of file ReBuffer.hh.

References lCount.

5.33.3.9 void clReBuffer::Clear () `[inline]`

Clear FIFO contents and reset buffer size.

Definition at line 87 of file ReBuffer.hh.

References Buffer, clAlloc::Free(), lCount, lGetIndex, and lPutIndex.

Referenced by clFlipBand::Clear(), GetPtr(), and clFilter::Uninitialize().

5.33.3.10 clReBuffer & clReBuffer::operator= (const clReBuffer &)

Definition at line 220 of file ReBuffer.cc.

References Buffer, clAlloc::Copy(), lCount, lGetIndex, lPutIndex, and lSize.

5.33.4 Member Data Documentation

5.33.4.1 long clReBuffer::lSize [private]

Definition at line 38 of file ReBuffer.hh.

Referenced by operator=().

5.33.4.2 long clReBuffer::lPutIndex [private]

Definition at line 39 of file ReBuffer.hh.

Referenced by CheckSize(), Clear(), clReBuffer(), operator=(), and Put().

5.33.4.3 long clReBuffer::lGetIndex [private]

Definition at line 40 of file ReBuffer.hh.

Referenced by CheckSize(), Clear(), clReBuffer(), Get(), GetPtr(), Index(), and operator=().

5.33.4.4 long clReBuffer::lCount [private]

Definition at line 41 of file ReBuffer.hh.

Referenced by CheckSize(), Clear(), clReBuffer(), Get(), GetCount(), GetPtr(), operator=(), and Put().

5.33.4.5 clAlloc clReBuffer::Buffer [private]

Definition at line 42 of file ReBuffer.hh.

Referenced by CheckSize(), Clear(), Get(), GetPtr(), Index(), operator=(), and Put().

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

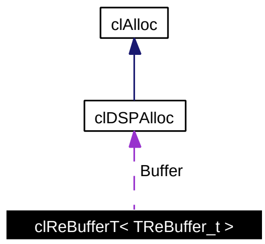
- ReBuffer.hh
- ReBuffer.cc

5.34 clReBufferT< TReBuffer_t > Class Template Reference

Template class for splitting data into buffers of different sizes.

`#include <ReBufferT.hh>`

Collaboration diagram for clReBufferT< TReBuffer_t >:



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.

Definition at line 54 of file ReBufferT.hh.

5.34.2 Constructor & Destructor Documentation

5.34.2.1 template<class TReBuffer_t> clReBufferT< TReBuffer_t >::clReBufferT () [inline]

Definition at line 131 of file ReBufferT.hh.

5.34.2.2 template<class TReBuffer_t> clReBufferT< TReBuffer_t >::clReBufferT (const clReBufferT< TReBuffer_t > & *CopySrc*) [inline]

Definition at line 138 of file ReBufferT.hh.

5.34.2.3 template<class TReBuffer_t> clReBufferT< TReBuffer_t >::clReBufferT (long *lNewSize*) [inline]

Definition at line 146 of file ReBufferT.hh.

5.34.2.4 template<class TReBuffer_t> clReBufferT< TReBuffer_t >::clReBufferT (const TReBuffer_t * *fpSrcData*, long *lSrcCount*) [inline]

Definition at line 154 of file ReBufferT.hh.

5.34.2.5 template<class TReBuffer_t> clReBufferT< TReBuffer_t >::~clReBufferT () [inline]

Definition at line 162 of file ReBufferT.hh.

5.34.3 Member Function Documentation

5.34.3.1 template<class TReBuffer_t> void clReBufferT< TReBuffer_t >::CheckSize (long *lDataCount*) [inline, private]

Definition at line 62 of file ReBufferT.hh.

Referenced by clReBufferT< TDSPVector_t >::Put(), clReBufferT< TDSPVector_t >::Resize(), and clReBufferT< TDSPVector_t >::SetSize().

5.34.3.2 template<class TReBuffer_t> TReBuffer_t* clReBufferT< TReBuffer_t >::GetPtr () [inline, protected]

Reimplemented in clDSPVector< TDSPVector_t >.

Definition at line 100 of file ReBufferT.hh.

Referenced by clDSPVector< TDSPVector_t >::GetPtr(), and clReBufferT< TDSPVector_t >::Put().

5.34.3.3 template<class TReBuffer_t> void clReBufferT< TReBuffer_t >::CopyGet (TReBuffer_t * *fpDestData*, long *lDestCount*) const [inline, protected]

Definition at line 111 of file ReBufferT.hh.

Referenced by clReBufferT< TDSPVector_t >::operator=().

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.

Definition at line 170 of file ReBufferT.hh.

Referenced by clReBufferT< TDSPVector_t >::clReBufferT(), clReBufferT< TDSPVector_t >::GetPtr(), and clReBufferT< TDSPVector_t >::Put().

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

Definition at line 198 of file ReBufferT.hh.

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

Definition at line 211 of file ReBufferT.hh.

Referenced by clReBufferT< TDSPVector_t >::GetPtr().

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

Definition at line 240 of file ReBufferT.hh.

5.34.3.8 template<class TReBuffer_t> long clReBufferT< TReBuffer_t >::Size () const [inline]

Definition at line 244 of file ReBufferT.hh.

Referenced by clReBufferT< TDSPVector_t >::operator=(), clReBufferT< TDSPVector_t >::Put(), and clDSPVector< TDSPVector_t >::Size().

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.

Reimplemented in clDSPVector< TDSPVector_t >.

Definition at line 253 of file ReBufferT.hh.

Referenced by clReBufferT< TDSPVector_t >::clReBufferT(), clReBufferT< TDSPVector_t >::operator=(), and clDSPVector< TDSPVector_t >::SetSize().

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.

Definition at line 265 of file ReBufferT.hh.

5.34.3.11 template<class TReBuffer_t> void clReBufferT< TReBuffer_t >::Clear () [inline]

Clear FIFO contents and reset buffer size.

Definition at line 275 of file ReBufferT.hh.

Referenced by clReBufferT< TDSPVector_t >::GetPtr(), and clReBufferT< TDSPVector_t >::SetSize().

5.34.3.12 template<class TReBuffer_t> clReBufferT& clReBufferT< TReBuffer_t >::operator= (const clReBufferT< TReBuffer_t > & *Src*) [inline]

Definition at line 283 of file ReBufferT.hh.

5.34.3.13]

template<class TReBuffer_t> TReBuffer_t& clReBufferT< TReBuffer_t >::operator[ ] (long *lIndex*) [inline]

Definition at line 291 of file ReBufferT.hh.

5.34.4 Member Data Documentation

5.34.4.1 template<class TReBuffer_t> long clReBufferT< TReBuffer_t >::lSize [private]

Definition at line 56 of file ReBufferT.hh.

5.34.4.2 template<class TReBuffer_t> long clReBufferT< TReBuffer_t >::lPutIndex [private]

Definition at line 57 of file ReBufferT.hh.

5.34.4.3 template<class TReBuffer_t> long clReBufferT< TReBuffer_t >::lGetIndex [private]

Definition at line 58 of file ReBufferT.hh.

5.34.4.4 template<class TReBuffer_t> long clReBufferT< TReBuffer_t >::lCount [private]

Definition at line 59 of file ReBufferT.hh.

5.34.4.5 template<class TReBuffer_t> clDSPAlloc clReBufferT< TReBuffer_t >::Buffer [private]

Definition at line 60 of file ReBufferT.hh.

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

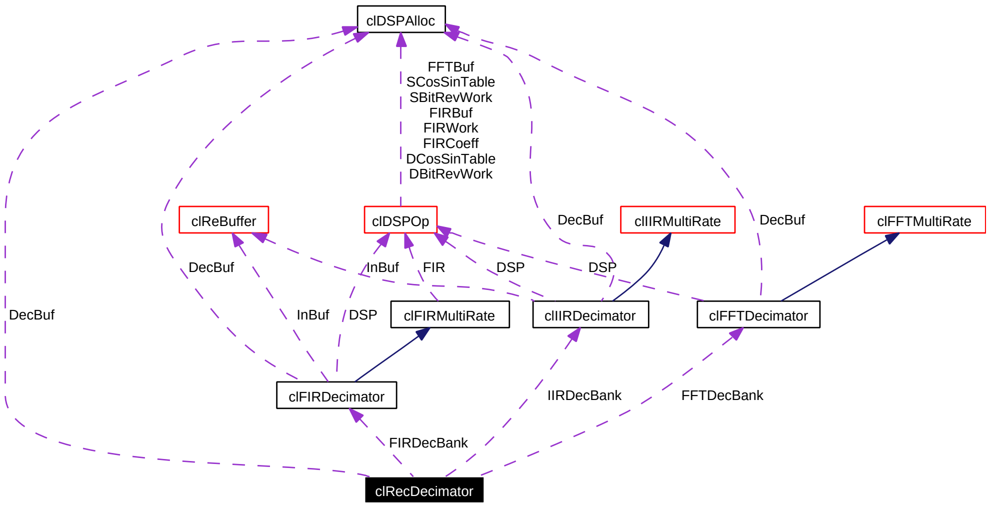
- ReBufferT.hh

5.35 clRecDecimator Class Reference

Recursive decimation filter class implementation.

`#include <RecDecimator.hh>`

Collaboration diagram for clRecDecimator:



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.

Definition at line 42 of file RecDecimator.hh.

5.35.2 Member Enumeration Documentation

5.35.2.1 enum clRecDecimator::eFilterType

Enumeration values:

FILTER_TYPE_FFT

FILTER_TYPE_FIR

FILTER_TYPE_IIR

Definition at line 57 of file RecDecimator.hh.

5.35.3 Constructor & Destructor Documentation

5.35.3.1 clRecDecimator::clRecDecimator ()

Definition at line 59 of file RecDecimator.cc.

References FILTER_TYPE_FFT, iType, and lSubRounds.

5.35.3.2 clRecDecimator::~clRecDecimator ()

Definition at line 67 of file RecDecimator.cc.

References Uninitialize().

5.35.4 Member Function Documentation

5.35.4.1 void clRecDecimator::InitHalves (double) `[private]`

Definition at line 34 of file RecDecimator.cc.

References bpHalves, and lSubRounds.

Referenced by Initialize().

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

Definition at line 73 of file RecDecimator.cc.

References bpHalves, DSP_FILT_DEF_OVERLAPF, FFTDecBank, FILTER_-TYPE_FFT, FILTER_TYPE_FIR, FILTER_TYPE_IIR, FIRDecBank, IIRDecBank, InitHalves(), clIIRMultiRate::Initialize(), clFIRMultiRate::Initialize(), clFFTMulti-Rate::Initialize(), iType, lDecSize, lSubRounds, RECDEC_MAX_SUB_ROUNDS, cl-Alloc::Size(), and Uninitialize().

Referenced by Initialize().

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.

Definition at line 110 of file RecDecimator.cc.

References bpHalves, DSP_FILT_DEF_OVERLAP, FFTDecBank, FILTER_-TYPE_FFT, FILTER_TYPE_FIR, FILTER_TYPE_IIR, FIRDecBank, IIRDecBank, InitHalves(), clIIRMultiRate::Initialize(), clFIRMultiRate::Initialize(), clFFTMulti-Rate::Initialize(), iType, lDecSize, lSubRounds, RECDEC_MAX_SUB_ROUNDS, clAlloc::Size(), and Uninitialize().

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.

Definition at line 147 of file RecDecimator.cc.

References FILTER_TYPE_FFT, FILTER_TYPE_FIR, and Initialize().

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.

Definition at line 159 of file RecDecimator.cc.

References FILTER_TYPE_FFT, FILTER_TYPE_FIR, and Initialize().

5.35.4.6 void clRecDecimator::Uninitialize ()

Uninitialize decimator.

Definition at line 171 of file RecDecimator.cc.

References FFTDecBank, FILTER_TYPE_FFT, FILTER_TYPE_FIR, FILTER_-TYPE_IIR, FIRDecBank, clAlloc::Free(), IIRDecBank, iType, lSubRounds, clIIRDecimator::Uninitialize(), clFIRDecimator::Uninitialize(), and cl-FFTDecimator::Uninitialize().

Referenced by Initialize(), and ∼clRecDecimator().

5.35.4.7 void clRecDecimator::Put (const float ∗, long)

Feed data into decimator.

Parameters:

SrcData Source data

SrcCount Source data count

Definition at line 195 of file RecDecimator.cc.

References FFTDecBank, FILTER_TYPE_FFT, FILTER_TYPE_FIR, FILTER_-TYPE_IIR, FIRDecBank, clIIRDecimator::Get(), clFIRDecimator::Get(), clFFTDecimator::Get(), IIRDecBank, iType, lDecSize, lSubRounds, cl-IIRDecimator::Put(), clFIRDecimator::Put(), and clFFTDecimator::Put().

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.

Definition at line 234 of file RecDecimator.cc.

References FFTDecBank, FILTER_TYPE_FFT, FILTER_TYPE_FIR, FILTER_-TYPE_IIR, FIRDecBank, clIIRDecimator::Get(), clFIRDecimator::Get(), clFFTDecimator::Get(), IIRDecBank, iType, lDecSize, lSubRounds, cl-IIRDecimator::Put(), clFIRDecimator::Put(), and clFFTDecimator::Put().

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?

Definition at line 273 of file RecDecimator.cc.

References FFTDecBank, FILTER_TYPE_FFT, FILTER_TYPE_FIR, FILTER_-TYPE_IIR, FIRDecBank, clIIRDecimator::Get(), clFIRDecimator::Get(), cl-FFTDecimator::Get(), IIRDecBank, iType, and lSubRounds.

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.

Definition at line 288 of file RecDecimator.cc.

References FFTDecBank, FILTER_TYPE_FFT, FILTER_TYPE_FIR, FILTER_-TYPE_IIR, FIRDecBank, clIIRDecimator::Get(), clFIRDecimator::Get(), cl-FFTDecimator::Get(), IIRDecBank, iType, and lSubRounds.

5.35.5 Member Data Documentation

5.35.5.1 bool clRecDecimator::bInitialized `[private]`

Definition at line 44 of file RecDecimator.hh.

5.35.5.2 int clRecDecimator::iType `[private]`

Type of filter; 0 = FFT, 1 = FIR, 2 = IIR.

Definition at line 45 of file RecDecimator.hh.

Referenced by clRecDecimator(), Get(), Initialize(), Put(), and Uninitialize().

5.35.5.3 long clRecDecimator::lFactor `[private]`

Decimation factor.

Definition at line 46 of file RecDecimator.hh.

5.35.5.4 long clRecDecimator::lFilterSize [private]

Size of input buffer.

Definition at line 47 of file RecDecimator.hh.

5.35.5.5 long clRecDecimator::lSubRounds [private]

Number of recursive rounds.

Definition at line 48 of file RecDecimator.hh.

Referenced by clRecDecimator(), Get(), InitHalves(), Initialize(), Put(), and Uninitialize().

5.35.5.6 long clRecDecimator::lDecSize [private]

Size of recursion buffer.

Definition at line 49 of file RecDecimator.hh.

Referenced by Initialize(), and Put().

5.35.5.7 bool clRecDecimator::bpHalves[RECDEC_MAX_SUB_ROUNDS] [private]

Definition at line 50 of file RecDecimator.hh.

Referenced by InitHalves(), and Initialize().

5.35.5.8 clDSPAlloc clRecDecimator::DecBuf [private]

Definition at line 51 of file RecDecimator.hh.

5.35.5.9 clFFTDecimator clRecDecimator::FFTDecBank[RECDEC_MAX_-SUB_ROUNDS] [private]

Definition at line 52 of file RecDecimator.hh.

Referenced by Get(), Initialize(), Put(), and Uninitialize().

5.35.5.10 clFIRDecimator clRecDecimator::FIRDecBank[RECDEC_MAX_-SUB_ROUNDS] [private]

Definition at line 53 of file RecDecimator.hh.

Referenced by Get(), Initialize(), Put(), and Uninitialize().

5.35.5.11 clIIRDecimator clRecDecimator::IIRDecBank[RECDEC_MAX_-SUB_ROUNDS] [private]

Definition at line 54 of file RecDecimator.hh.

Referenced by Get(), Initialize(), Put(), and Uninitialize().

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

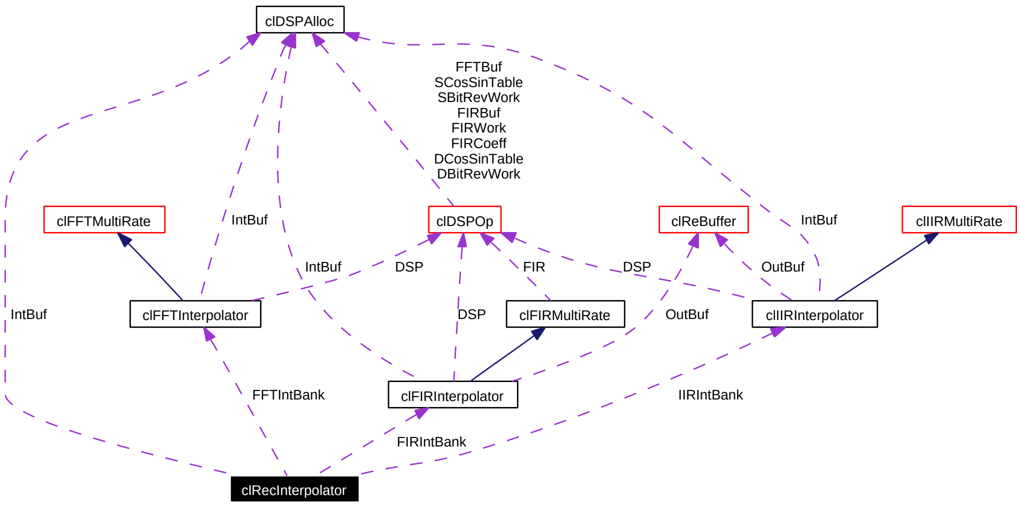
- RecDecimator.hh
- RecDecimator.cc

5.36 clRecInterpolator Class Reference

Recursive interpolation filter class implementation.

`#include <RecInterpolator.hh>`

Collaboration diagram for clRecInterpolator:



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.

Definition at line 42 of file RecInterpolator.hh.

5.36.2 Member Enumeration Documentation

5.36.2.1 enum clRecInterpolator::eFilterType

Enumeration values:

FILTER_TYPE_FFT

FILTER_TYPE_FIR

FILTER_TYPE_IIR

Definition at line 57 of file RecInterpolator.hh.

5.36.3 Constructor & Destructor Documentation

5.36.3.1 clRecInterpolator::clRecInterpolator ()

Definition at line 59 of file RecInterpolator.cc.

5.36.3.2 clRecInterpolator::∼clRecInterpolator ()

Definition at line 67 of file RecInterpolator.cc.

References Uninitialize().

5.36.4 Member Function Documentation

5.36.4.1 void clRecInterpolator::InitHalves (double) `[private]`

Definition at line 34 of file RecInterpolator.cc.

Referenced by Initialize().

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

Definition at line 73 of file RecInterpolator.cc.

References DSP_FILT_DEF_OVERLAPF, FFTIntBank, FIRIntBank, IIRIntBank, InitHalves(), clIIRMultiRate::Initialize(), clFIRMultiRate::Initialize(), clFFTMultiRate::Initialize(), lIntSize, RECINT_MAX_SUB_ROUNDS, clAlloc::Size(), and Uninitialize().

Referenced by Initialize().

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.

Definition at line 110 of file RecInterpolator.cc.

References DSP_FILT_DEF_OVERLAP, FFTIntBank, FIRIntBank, IIRIntBank, InitHalves(), clIIRMultiRate::Initialize(), clFIRMultiRate::Initialize(), clFFTMultiRate::Initialize(), lIntSize, RECINT_MAX_SUB_ROUNDS, clAlloc::Size(), and Uninitialize().

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.

Definition at line 147 of file RecInterpolator.cc.

References Initialize().

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.

Definition at line 159 of file RecInterpolator.cc.

References Initialize().

5.36.4.6 void clRecInterpolator::Uninitialize ()

Uninitialize interpolator.

Definition at line 171 of file RecInterpolator.cc.

References FFTIntBank, FIRIntBank, clAlloc::Free(), IIRIntBank, clIIRInterpolator::Uninitialize(), clFIRInterpolator::Uninitialize(), and clFFTInterpolator::Uninitialize().

Referenced by Initialize(), and ~clRecInterpolator().

5.36.4.7 void clRecInterpolator::Put (const float ∗, long)

Feed data into interpolator.

Parameters:

SrcData Source data

SrcCount Source data count

Definition at line 195 of file RecInterpolator.cc.

References FFTIntBank, FIRIntBank, clIIRInterpolator::Get(), clFIRInterpolator::Get(), clFFTInterpolator::Get(), IIRIntBank, lIntSize, clIIRInterpolator::Put(), clFIRInterpolator::Put(), and clFFTInterpolator::Put().

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.

Definition at line 234 of file RecInterpolator.cc.

References FFTIntBank, FIRIntBank, clIIRInterpolator::Get(), clFIRInterpolator::Get(), clFFTInterpolator::Get(), IIRIntBank, lIntSize, clIIRInterpolator::Put(), clFIRInterpolator::Put(), and clFFTInterpolator::Put().

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?

Definition at line 273 of file RecInterpolator.cc.

References FFTIntBank, FIRIntBank, clIIRInterpolator::Get(), clFIRInterpolator::Get(), clFFTInterpolator::Get(), and IIRIntBank.

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.

Definition at line 288 of file RecInterpolator.cc.

References FFTIntBank, FIRIntBank, clIIRInterpolator::Get(), clFIRInterpolator::Get(), clFFTInterpolator::Get(), and IIRIntBank.

5.36.5 Member Data Documentation

5.36.5.1 bool clRecInterpolator::bInitialized `[private]`

Definition at line 44 of file RecInterpolator.hh.

5.36.5.2 int clRecInterpolator::iType `[private]`

Type filter; 0 = FFT, 1= FIR, 2 = IIR.

Definition at line 45 of file RecInterpolator.hh.

5.36.5.3 long clRecInterpolator::lFactor `[private]`

Interpolation factor.

Definition at line 46 of file RecInterpolator.hh.

5.36.5.4 long clRecInterpolator::lFilterSize `[private]`

Size of input buffer.

Definition at line 47 of file RecInterpolator.hh.

5.36.5.5 long clRecInterpolator::lSubRounds `[private]`

Number of recursive rounds.

Definition at line 48 of file RecInterpolator.hh.

5.36.5.6 long clRecInterpolator::lIntSize [private]

Size of recursion buffer.

Definition at line 49 of file RecInterpolator.hh.

Referenced by Initialize(), and Put().

5.36.5.7 bool clRecInterpolator::bpHalves[RECINT_MAX_SUB_ROUNDS] [private]

Definition at line 50 of file RecInterpolator.hh.

5.36.5.8 clDSPAlloc clRecInterpolator::IntBuf [private]

Definition at line 51 of file RecInterpolator.hh.

5.36.5.9 clFFTInterpolator clRecInterpolator::FFTIntBank[RECINT_MAX_-SUB_ROUNDS] [private]

Definition at line 52 of file RecInterpolator.hh.

Referenced by Get(), Initialize(), Put(), and Uninitialize().

5.36.5.10 clFIRInterpolator clRecInterpolator::FIRIntBank[RECINT_MAX_-SUB_ROUNDS] [private]

Definition at line 53 of file RecInterpolator.hh.

Referenced by Get(), Initialize(), Put(), and Uninitialize().

5.36.5.11 clIIRInterpolator clRecInterpolator::IIRIntBank[RECINT_MAX_-SUB_ROUNDS] [private]

Definition at line 54 of file RecInterpolator.hh.

Referenced by Get(), Initialize(), Put(), and Uninitialize().

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

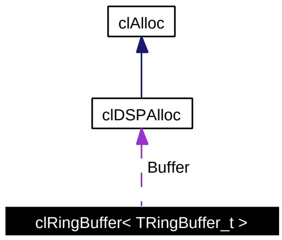
- RecInterpolator.hh
- RecInterpolator.cc

5.37 clRingBuffer< TRingBuffer_t > Class Template Reference

Class for predefined size lockless ring buffer.

`#include <RingBuffer.hh>`

Collaboration diagram for clRingBuffer< TRingBuffer_t >:



Public Member Functions

- clRingBuffer ()
- clRingBuffer (const clRingBuffer< TRingBuffer_t > &CopySrc)
- clRingBuffer (long lNewSize)
- void SetSize (long lNewSize)
- void Lock ()
- void UnLock ()
- long Size () const
- long GetCount () const
- void Put (const TRingBuffer_t ∗Src, long lSrcCount)
- bool Get (TRingBuffer_t ∗Dst, long lDstCount)

Private Attributes

- volatile long lSize
- volatile long lCount
- volatile long lPutIndex
- volatile long lGetIndex
- clDSPAlloc Buffer

5.37.1 Detailed Description

template<class TRingBuffer_t> class clRingBuffer< TRingBuffer_t >

Class for predefined size lockless ring buffer.

Definition at line 49 of file RingBuffer.hh.

5.37.2 Constructor & Destructor Documentation

5.37.2.1 template<class TRingBuffer_t> clRingBuffer< TRingBuffer_t >::clRingBuffer () `[inline]`

Definition at line 57 of file RingBuffer.hh.

5.37.2.2 template<class TRingBuffer_t> clRingBuffer< TRingBuffer_t >::clRingBuffer (const clRingBuffer< TRingBuffer_t > & *CopySrc*) `[inline]`

Definition at line 64 of file RingBuffer.hh.

References clRingBuffer< TRingBuffer_t >::Buffer, clRingBuffer< TRingBuffer_t >::lCount, clRingBuffer< TRingBuffer_t >::lGetIndex, clRingBuffer< TRingBuffer_t >::lPutIndex, and clRingBuffer< TRingBuffer_t >::lSize.

5.37.2.3 template<class TRingBuffer_t> clRingBuffer< TRingBuffer_t >::clRingBuffer (long *lNewSize*) `[inline]`

Definition at line 72 of file RingBuffer.hh.

References clRingBuffer< TRingBuffer_t >::SetSize().

5.37.3 Member Function Documentation

5.37.3.1 template<class TRingBuffer_t> void clRingBuffer< TRingBuffer_t >::SetSize (long *lNewSize*) `[inline]`

Definition at line 74 of file RingBuffer.hh.

References clAlloc::Size().

Referenced by clRingBuffer< TRingBuffer_t >::clRingBuffer().

5.37.3.2 template<class TRingBuffer_t> void clRingBuffer< TRingBuffer_t >::Lock () [inline]

Definition at line 82 of file RingBuffer.hh.

References clAlloc::Lock().

5.37.3.3 template<class TRingBuffer_t> void clRingBuffer< TRingBuffer_t >::UnLock () [inline]

Definition at line 84 of file RingBuffer.hh.

References clAlloc::UnLock().

5.37.3.4 template<class TRingBuffer_t> long clRingBuffer< TRingBuffer_t >::Size () const [inline]

Definition at line 86 of file RingBuffer.hh.

5.37.3.5 template<class TRingBuffer_t> long clRingBuffer< TRingBuffer_t >::GetCount () const [inline]

Definition at line 90 of file RingBuffer.hh.

5.37.3.6 template<class TRingBuffer_t> void clRingBuffer< TRingBuffer_t >::Put (const TRingBuffer_t * *Src*, long *lSrcCount*) [inline]

Definition at line 94 of file RingBuffer.hh.

References clDSPOp::Copy().

5.37.3.7 template<class TRingBuffer_t> bool clRingBuffer< TRingBuffer_t >::Get (TRingBuffer_t * *Dst*, long *lDstCount*) [inline]

Definition at line 118 of file RingBuffer.hh.

References clDSPOp::Copy().

5.37.4 Member Data Documentation

5.37.4.1 template<class TRingBuffer_t> volatile long clRingBuffer< TRingBuffer_t >::lSize [private]

Definition at line 51 of file RingBuffer.hh.

Referenced by clRingBuffer< TRingBuffer_t >::clRingBuffer().

5.37.4.2 template<class TRingBuffer_t> volatile long clRingBuffer< TRingBuffer_t >::lCount [private]

Definition at line 52 of file RingBuffer.hh.

Referenced by clRingBuffer< TRingBuffer_t >::clRingBuffer().

5.37.4.3 template<class TRingBuffer_t> volatile long clRingBuffer< TRingBuffer_t >::lPutIndex [private]

Definition at line 53 of file RingBuffer.hh.

Referenced by clRingBuffer< TRingBuffer_t >::clRingBuffer().

5.37.4.4 template<class TRingBuffer_t> volatile long clRingBuffer< TRingBuffer_t >::lGetIndex [private]

Definition at line 54 of file RingBuffer.hh.

Referenced by clRingBuffer< TRingBuffer_t >::clRingBuffer().

5.37.4.5 template<class TRingBuffer_t> clDSPAlloc clRingBuffer< TRingBuffer_t >::Buffer [private]

Definition at line 55 of file RingBuffer.hh.

Referenced by clRingBuffer< TRingBuffer_t >::clRingBuffer().

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

- RingBuffer.hh

5.38 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 ()
  Desctructor; 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

5.38.1 Detailed Description

Class implementation of SUSv2 read-write locks.

Definition at line 47 of file RWLock.hh.

5.38.2 Constructor & Destructor Documentation

5.38.2.1 clRWLock::clRWLock () `[inline]`

Constructor; creates and initializes the read-write lock.

Definition at line 54 of file RWLock.hh.

References ptrwlLock.

5.38.2.2 clRWLock::~clRWLock () `[inline]`

Desctructor; destroys the read-write lock.

Definition at line 59 of file RWLock.hh.

References ptrwlLock.

5.38.3 Member Function Documentation

5.38.3.1 bool clRWLock::WaitRead () `[inline]`

Obtain read-lock.

There can be multiple read-locks at the same time.

Returns:
Success

Definition at line 67 of file RWLock.hh.

References ptrwlLock.

5.38.3.2 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

Definition at line 80 of file RWLock.hh.

References ptrwlLock.

5.38.3.3 bool clRWLock::Release () `[inline]`

Release lock.

Returns:
Success

Definition at line 91 of file RWLock.hh.

References ptrwlLock.

5.38.3.4 bool clRWLock::TryRead () `[inline]`

Try to obtain read-lock (non-blocking).

Returns:
Success

Definition at line 102 of file RWLock.hh.

References ptrwlLock.

5.38.3.5 bool clRWLock::TryWrite () `[inline]`

Try to obtain exclusive write-lock (non-blocking).

Returns:
Success

Definition at line 113 of file RWLock.hh.

References ptrwlLock.

5.38.4 Member Data Documentation

5.38.4.1 pthread_rwlock_t clRWLock::ptrwlLock `[private]`

Definition at line 49 of file RWLock.hh.

Referenced by clRWLock(), Release(), TryRead(), TryWrite(), WaitRead(), WaitWrite(), and ~clRWLock().

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

- RWLock.hh

5.39 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

5.39.1 Detailed Description

Class implementation of POSIX counting semaphores.

Definition at line 37 of file Semaphore.hh.

5.39.2 Constructor & Destructor Documentation

5.39.2.1 clSemaphore::clSemaphore () `[inline]`

Constructor; create and initialize the semaphore.

Semaphore value is initialized to zero.

Definition at line 49 of file Semaphore.hh.

References semSemaphore.

5.39.2.2 clSemaphore::clSemaphore (unsigned int *uiSemValue*) `[inline]`

Constructor; create and initialize the semaphore.

Semaphore is initialized to caller specified value.

Parameters:

uiSemValue Initial semaphore value

Definition at line 61 of file Semaphore.hh.

References semSemaphore.

5.39.2.3 clSemaphore::~clSemaphore () `[inline]`

Destructor; destroy the semaphore.

Definition at line 70 of file Semaphore.hh.

References semSemaphore.

5.39.3 Member Function Documentation

5.39.3.1 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

Definition at line 83 of file Semaphore.hh.

References semSemaphore.

5.39.3.2 void clSemaphore::Wait () `[inline]`

Wait until semaphore count becomes non-zero and then decrement the count.

Definition at line 100 of file Semaphore.hh.

References semSemaphore.

5.39.3.3 bool clSemaphore::TryWait () `[inline]`

Test if semaphore count is non-zero, if it is, then count is decremented.

Definition at line 113 of file Semaphore.hh.

References semSemaphore.

5.39.3.4 bool clSemaphore::Post () `[inline]`

Post (increment) semaphore.

Returns:
Success

Definition at line 131 of file Semaphore.hh.

References semSemaphore.

5.39.3.5 int clSemaphore::GetValue () `[inline]`

Get value of semaphore.

Returns:
Semaphore count

Definition at line 148 of file Semaphore.hh.

References semSemaphore.

5.39.4 Member Data Documentation

5.39.4.1 sem_t clSemaphore::semSemaphore [private]

Definition at line 40 of file Semaphore.hh.

Referenced by clSemaphore(), GetValue(), Initialize(), Post(), TryWait(), Wait(), and ∼clSemaphore().

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

- Semaphore.hh

5.40 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 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 transform.

Author:
Takuya OOURA
Jussi Laako

Note:
DFT parts of this class has been wrapped into clDSPOp with easier interface.

Definition at line 65 of file Transform4.hh.

5.40.2 Member Function Documentation

5.40.2.1 void clTransform4::makewt (long, long ∗, float ∗) `[private]`

Definition at line 653 of file Transform4.cc.

References bitrv2().

Referenced by cdft(), ddct(), ddst(), dfct(), dfst(), and rdft().

5.40.2.2 void clTransform4::makewt (long, long ∗, double ∗) `[private]`

Definition at line 1700 of file Transform4.cc.

References bitrv2().

5.40.2.3 void clTransform4::makect (long, long ∗, float ∗) [private]

Definition at line 695 of file Transform4.cc.

Referenced by ddct(), ddst(), dfct(), dfst(), and rdft().

5.40.2.4 void clTransform4::makect (long, long ∗, double ∗) [private]

Definition at line 1742 of file Transform4.cc.

5.40.2.5 T4_INLINE void clTransform4::bitrv2 (long, long ∗, float ∗) [private]

Definition at line 727 of file Transform4.cc.

References T4_INLINE.

Referenced by cdft(), ddct(), ddst(), dfct(), dfst(), makewt(), and rdft().

5.40.2.6 T4_INLINE void clTransform4::bitrv2 (long, long ∗, double ∗) [private]

Definition at line 1774 of file Transform4.cc.

References T4_INLINE.

5.40.2.7 T4_INLINE void clTransform4::bitrv2conj (long, long ∗, float ∗) [private]

Definition at line 827 of file Transform4.cc.

References T4_INLINE.

Referenced by cdft().

5.40.2.8 T4_INLINE void clTransform4::bitrv2conj (long, long ∗, double ∗) [private]

Definition at line 1874 of file Transform4.cc.

References T4_INLINE.

5.40.2.9 T4_INLINE void clTransform4::cftfsub (long, float ∗, float ∗) [private]

Definition at line 936 of file Transform4.cc.

References cft1st(), cftmdl(), and T4_INLINE.

Referenced by cdft(), ddct(), ddst(), dfct(), dfst(), and rdft().

5.40.2.10 T4_INLINE void clTransform4::cftfsub (long, double ∗, double ∗) [private]

Definition at line 1983 of file Transform4.cc.

References cft1st(), cftmdl(), and T4_INLINE.

5.40.2.11 T4_INLINE void clTransform4::cftbsub (long, float ∗, float ∗) [private]

Definition at line 986 of file Transform4.cc.

References cft1st(), cftmdl(), and T4_INLINE.

Referenced by cdft(), ddct(), ddst(), and rdft().

5.40.2.12 T4_INLINE void clTransform4::cftbsub (long, double ∗, double ∗) [private]

Definition at line 2033 of file Transform4.cc.

References cft1st(), cftmdl(), and T4_INLINE.

5.40.2.13 T4_INLINE void clTransform4::cft1st (long, float ∗, float ∗) [private]

Definition at line 1036 of file Transform4.cc.

References T4_INLINE.

Referenced by cftbsub(), and cftfsub().

5.40.2.14 T4_INLINE void clTransform4::cft1st (long, double ∗, double ∗) [private]

Definition at line 2083 of file Transform4.cc.

References T4_INLINE.

5.40.2.15 T4_INLINE void clTransform4::cftmdl (long, long, float ∗, float ∗) [private]

Definition at line 1141 of file Transform4.cc.

References T4_INLINE.

Referenced by cftbsub(), and cftfsub().

5.40.2.16 T4_INLINE void clTransform4::cftmdl (long, long, double *, double *) [private]

Definition at line 2188 of file Transform4.cc.

References T4_INLINE.

5.40.2.17 T4_INLINE void clTransform4::rftfsub (long, float *, long, float *) [private]

Definition at line 1268 of file Transform4.cc.

References T4_INLINE.

Referenced by ddct(), ddst(), dfct(), dfst(), and rdft().

5.40.2.18 T4_INLINE void clTransform4::rftfsub (long, double *, long, double *) [private]

Definition at line 2315 of file Transform4.cc.

References T4_INLINE.

5.40.2.19 T4_INLINE void clTransform4::rftbsub (long, float *, long, float *) [private]

Definition at line 1293 of file Transform4.cc.

References T4_INLINE.

Referenced by ddct(), ddst(), and rdft().

5.40.2.20 T4_INLINE void clTransform4::rftbsub (long, double *, long, double *) [private]

Definition at line 2340 of file Transform4.cc.

References T4_INLINE.

5.40.2.21 T4_INLINE void clTransform4::dctsub (long, float *, long, float *) [private]

Definition at line 1320 of file Transform4.cc.

References T4_INLINE.

Referenced by ddct(), and dfct().

5.40.2.22 T4_INLINE void clTransform4::dctsub (long, double ∗, long, double ∗) [private]

Definition at line 2367 of file Transform4.cc.

References T4_INLINE.

5.40.2.23 T4_INLINE void clTransform4::dstsub (long, float ∗, long, float ∗) [private]

Definition at line 1341 of file Transform4.cc.

References T4_INLINE.

Referenced by ddst(), and dfst().

5.40.2.24 T4_INLINE void clTransform4::dstsub (long, double ∗, long, double ∗) [private]

Definition at line 2388 of file Transform4.cc.

References T4_INLINE.

5.40.2.25 void clTransform4::cdft (long, long, float ∗, long ∗, float ∗)

Complex DFT.

Forward:

$$X(k) = \sum_{j=0}^{N-1} x(j) \exp(2\pi ijk/N), 0 \leq k \leq N-1$$

Inverse:

$$X(k) = \sum_{j=0}^{N-1} x(j) \exp(-2\pi ijk/N), 0 \leq k \leq N-1$$

Input data:

$$\left| \begin{array}{l} a(2j) = \Re_x(j) \\ a(2j+1) = \Im_x(j) \end{array} \right. ,0 \leq j \leq N-1$$

Output data:

$$\left| \begin{array}{l} a(2k) = \Re_X(k) \\ a(2k+1) = \Im_X(k) \end{array} \right. ,0 \leq k \leq N-1$$

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

Definition at line 318 of file Transform4.cc.

References bitrv2(), bitrv2conj(), cftbsub(), cftfsub(), and makewt().

5.40.2.26 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.

Definition at line 1365 of file Transform4.cc.

References bitrv2(), bitrv2conj(), cftbsub(), cftfsub(), and makewt().

5.40.2.27 void clTransform4::rdft (long, long, float ∗, long ∗, float ∗)

Real DFT.

Forward:

$$\left| \begin{array}{ll} \Re_X(k) = \sum_{j=0}^{N-1} a(j)\cos(2\pi jk/N) & ,0 \le k \le N/2 \\ \Im_X(k) = \sum_{j=0}^{N-1} a(j)\sin(2\pi jk/N) & ,0 < k < N/2 \end{array} \right.$$

Inverse:

$$a(k) = \frac{\left(\Re_X(0) + \Re_X(\frac{N}{2})cos(\pi k)\right)}{2} + \sum_{j=1}^{N/2-1} \Re_X(j)\cos(2\pi jk/N) + \sum_{j=1}^{N/2-1} \Im_X(j)\sin(2\pi jk/N), 0 \le k \le N-1$$

Output data:

$$\left| \begin{array}{ll} a(2k) = \Re_X(k) & ,0 \le k < N/2 \\ a(2k+1) = \Im_X(k) & ,0 < k < N/2 \\ a(1) = \Re_X(N/2) & \end{array} \right.$$

Input data:

$$\left| \begin{array}{ll} a(2j) = \Re_X(j) & ,0 \le j < N/2 \\ a(2j+1) = \Im_X(j) & ,0 < j < N/2 \\ a(1) = \Re_X(N/2) & \end{array} \right.$$

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

Definition at line 337 of file Transform4.cc.

References bitrv2(), cftbsub(), cftfsub(), makect(), makewt(), rftbsub(), and rftfsub().

5.40.2.28 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.

Definition at line 1384 of file Transform4.cc.

References bitrv2(), cftbsub(), cftfsub(), makect(), makewt(), rftbsub(), and rftfsub().

5.40.2.29 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

Definition at line 377 of file Transform4.cc.

References bitrv2(), cftbsub(), cftfsub(), dctsub(), makect(), makewt(), rftbsub(), and rftfsub().

5.40.2.30 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.

Definition at line 1424 of file Transform4.cc.

References bitrv2(), cftbsub(), cftfsub(), dctsub(), makect(), makewt(), rftbsub(), and rftfsub().

5.40.2.31 void clTransform4::ddst (long, long, float ∗, long ∗, float ∗)

DST.

Forward:

$$S(k) = \sum_{j=1}^{N} A(j) \sin(\pi j(k + 1/2)/N), 0 \le k \le N - 1$$

Inverse:

$$S(k) = \sum_{j=0}^{N-1} a(j) \sin(\pi(j + 1/2)k/N), 0 < k \le N$$

Input data (forward):

$$\left| \begin{array}{ll} a(j) = A(j) & , 0 < j < N \\ a(0) = A(N) & \end{array} \right.$$

Output data (forward):

$$a(k) = S(k), 0 \le k \le N - 1$$

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

Definition at line 428 of file Transform4.cc.

References bitrv2(), cftbsub(), cftfsub(), dstsub(), makect(), makewt(), rftbsub(), and rftfsub().

5.40.2.32 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.

Definition at line 1475 of file Transform4.cc.

References bitrv2(), cftbsub(), cftfsub(), dstsub(), makect(), makewt(), rftbsub(), and rftfsub().

5.40.2.33 void clTransform4::dfct (long, float ∗, float ∗, long ∗, float ∗)

CT of RDFT.

Definition:

$$C(k) = \sum_{j=0}^{N} a(j)\cos(\pi jk/N), 0 \le k \le N$$

Output data:

$$a(k) = C(k), 0 \le k \le N$$

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

Definition at line 479 of file Transform4.cc.

References bitrv2(), cftfsub(), dctsub(), makect(), makewt(), and rftfsub().

5.40.2.34 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.

Definition at line 1526 of file Transform4.cc.

References bitrv2(), cftfsub(), dctsub(), makect(), makewt(), and rftfsub().

5.40.2.35 void clTransform4::dfst (long, float ∗, float ∗, long ∗, float ∗)

ST of RDFT.

Definition:

$$S(k) = \sum_{j=1}^{N-1} a(j)\sin(\pi jk/N), 0 < k < N$$

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

Definition at line 569 of file Transform4.cc.

References bitrv2(), cftfsub(), dstsub(), makect(), makewt(), and rftfsub().

5.40.2.36 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.

Definition at line 1616 of file Transform4.cc.

References bitrv2(), cftfsub(), dstsub(), makect(), makewt(), and rftfsub().

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

- Transform4.hh
- Transform4.cc

5.41 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 ∗)

5.41.1 Detailed Description

Decimation-in-frequency radix-2/4/8 transform.

Author:
Takuya OOURA
Jussi Laako

Note:
See clTransform4 for details.

Definition at line 64 of file Transform8.hh.

5.41.2 Member Function Documentation

5.41.2.1 void clTransform8::makewt (long, long ∗, float ∗) `[private]`

Definition at line 653 of file Transform8.cc.

References bitrv2().

Referenced by cdft(), ddct(), ddst(), dfct(), dfst(), and rdft().

5.41.2.2 void clTransform8::makewt (long, long ∗, double ∗) `[private]`

Definition at line 1997 of file Transform8.cc.

References bitrv2().

5.41.2.3 void clTransform8::makect (long, long ∗, float ∗) `[private]`

Definition at line 701 of file Transform8.cc.

Referenced by ddct(), ddst(), dfct(), dfst(), and rdft().

5.41.2.4 void clTransform8::makect (long, long ∗, double ∗) `[private]`

Definition at line 2045 of file Transform8.cc.

5.41.2.5 T8_INLINE void clTransform8::bitrv2 (long, long ∗, float ∗) [private]

Definition at line 733 of file Transform8.cc.

References T8_INLINE.

Referenced by cdft(), ddct(), ddst(), dfct(), dfst(), makewt(), and rdft().

5.41.2.6 T8_INLINE void clTransform8::bitrv2 (long, long ∗, double ∗) [private]

Definition at line 2077 of file Transform8.cc.

References T8_INLINE.

5.41.2.7 T8_INLINE void clTransform8::bitrv2conj (long, long ∗, float ∗) [private]

Definition at line 833 of file Transform8.cc.

References T8_INLINE.

Referenced by cdft().

5.41.2.8 T8_INLINE void clTransform8::bitrv2conj (long, long ∗, double ∗) [private]

Definition at line 2177 of file Transform8.cc.

References T8_INLINE.

5.41.2.9 T8_INLINE void clTransform8::cftfsub (long, float ∗, float ∗) [private]

Definition at line 942 of file Transform8.cc.

References cft1st(), cftmdl(), and T8_INLINE.

Referenced by cdft(), ddct(), ddst(), dfct(), dfst(), and rdft().

5.41.2.10 T8_INLINE void clTransform8::cftfsub (long, double ∗, double ∗) [private]

Definition at line 2286 of file Transform8.cc.

References cft1st(), cftmdl(), and T8_INLINE.

5.41.2.11 T8_INLINE void clTransform8::cftbsub (long, float *, float *) [private]

Definition at line 992 of file Transform8.cc.

References cft1st(), cftmdl(), and T8_INLINE.

Referenced by cdft(), ddct(), ddst(), and rdft().

5.41.2.12 T8_INLINE void clTransform8::cftbsub (long, double *, double *) [private]

Definition at line 2336 of file Transform8.cc.

References cft1st(), cftmdl(), and T8_INLINE.

5.41.2.13 T8_INLINE void clTransform8::cft1st (long, float *, float *) [private]

Definition at line 1107 of file Transform8.cc.

References T8_INLINE.

Referenced by cftbsub(), and cftfsub().

5.41.2.14 T8_INLINE void clTransform8::cft1st (long, double *, double *) [private]

Definition at line 2451 of file Transform8.cc.

References T8_INLINE.

5.41.2.15 T8_INLINE void clTransform8::cftmdl (long, long, float *, float *) [private]

Definition at line 1322 of file Transform8.cc.

References T8_INLINE.

Referenced by cftbsub(), and cftfsub().

5.41.2.16 T8_INLINE void clTransform8::cftmdl (long, long, double *, double *) [private]

Definition at line 2666 of file Transform8.cc.

References T8_INLINE.

5.41.2.17 T8_INLINE void clTransform8::rftfsub (long, float *, long, float *) [private]

Definition at line 1565 of file Transform8.cc.

References T8_INLINE.

Referenced by ddct(), ddst(), dfct(), dfst(), and rdft().

5.41.2.18 T8_INLINE void clTransform8::rftfsub (long, double *, long, double *) [private]

Definition at line 2909 of file Transform8.cc.

References T8_INLINE.

5.41.2.19 T8_INLINE void clTransform8::rftbsub (long, float *, long, float *) [private]

Definition at line 1590 of file Transform8.cc.

References T8_INLINE.

Referenced by ddct(), ddst(), and rdft().

5.41.2.20 T8_INLINE void clTransform8::rftbsub (long, double *, long, double *) [private]

Definition at line 2934 of file Transform8.cc.

References T8_INLINE.

5.41.2.21 T8_INLINE void clTransform8::dctsub (long, float *, long, float *) [private]

Definition at line 1617 of file Transform8.cc.

References T8_INLINE.

Referenced by ddct(), and dfct().

5.41.2.22 T8_INLINE void clTransform8::dctsub (long, double *, long, double *) [private]

Definition at line 2961 of file Transform8.cc.

References T8_INLINE.

5.41.2.23 T8_INLINE void clTransform8::dstsub (long, float ∗, long, float ∗) [private]

Definition at line 1638 of file Transform8.cc.

References T8_INLINE.

Referenced by ddst(), and dfst().

5.41.2.24 T8_INLINE void clTransform8::dstsub (long, double ∗, long, double ∗) [private]

Definition at line 2982 of file Transform8.cc.

References T8_INLINE.

5.41.2.25 void clTransform8::cdft (long, long, float ∗, long ∗, float ∗)

Definition at line 318 of file Transform8.cc.

References bitrv2(), bitrv2conj(), cftbsub(), cftfsub(), and makewt().

5.41.2.26 void clTransform8::cdft (long, long, double ∗, long ∗, double ∗)

Definition at line 1662 of file Transform8.cc.

References bitrv2(), bitrv2conj(), cftbsub(), cftfsub(), and makewt().

5.41.2.27 void clTransform8::rdft (long, long, float ∗, long ∗, float ∗)

Definition at line 337 of file Transform8.cc.

References bitrv2(), cftbsub(), cftfsub(), makect(), makewt(), rftbsub(), and rftfsub().

5.41.2.28 void clTransform8::rdft (long, long, double ∗, long ∗, double ∗)

Definition at line 1681 of file Transform8.cc.

References bitrv2(), cftbsub(), cftfsub(), makect(), makewt(), rftbsub(), and rftfsub().

5.41.2.29 void clTransform8::ddct (long, long, float ∗, long ∗, float ∗)

Definition at line 377 of file Transform8.cc.

References bitrv2(), cftbsub(), cftfsub(), dctsub(), makect(), makewt(), rftbsub(), and rftfsub().

5.41.2.30 void clTransform8::ddct (long, long, double ∗, long ∗, double ∗)

Definition at line 1721 of file Transform8.cc.

References bitrv2(), cftbsub(), cftfsub(), dctsub(), makect(), makewt(), rftbsub(), and rftfsub().

5.41.2.31 void clTransform8::ddst (long, long, float ∗, long ∗, float ∗)

Definition at line 428 of file Transform8.cc.

References bitrv2(), cftbsub(), cftfsub(), dstsub(), makect(), makewt(), rftbsub(), and rftfsub().

5.41.2.32 void clTransform8::ddst (long, long, double ∗, long ∗, double ∗)

Definition at line 1772 of file Transform8.cc.

References bitrv2(), cftbsub(), cftfsub(), dstsub(), makect(), makewt(), rftbsub(), and rftfsub().

5.41.2.33 void clTransform8::dfct (long, float ∗, float ∗, long ∗, float ∗)

Definition at line 479 of file Transform8.cc.

References bitrv2(), cftfsub(), dctsub(), makect(), makewt(), and rftfsub().

5.41.2.34 void clTransform8::dfct (long, double ∗, double ∗, long ∗, double ∗)

Definition at line 1823 of file Transform8.cc.

References bitrv2(), cftfsub(), dctsub(), makect(), makewt(), and rftfsub().

5.41.2.35 void clTransform8::dfst (long, float ∗, float ∗, long ∗, float ∗)

Definition at line 569 of file Transform8.cc.

References bitrv2(), cftfsub(), dstsub(), makect(), makewt(), and rftfsub().

5.41.2.36 void clTransform8::dfst (long, double ∗, double ∗, long ∗, double ∗)

Definition at line 1913 of file Transform8.cc.

References bitrv2(), cftfsub(), dstsub(), makect(), makewt(), and rftfsub().

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

- Transform8.hh
- Transform8.cc

5.42 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 ∗)

5.42.1 Detailed Description

Decimation-in-frequency split-radix transform.

Author:

Takuya OOURA
Jussi Laako

Note:

See clTransform4 for details.

Definition at line 92 of file TransformS.hh.

5.42.2 Constructor & Destructor Documentation

5.42.2.1 clTransformS::clTransformS () `[inline]`

Definition at line 160 of file TransformS.hh.

5.42.2.2 clTransformS::∼clTransformS () `[inline]`

Definition at line 161 of file TransformS.hh.

5.42.3 Member Function Documentation

5.42.3.1 void clTransformS::makeipt (long, long ∗) `[private]`

Definition at line 104 of file TransformS.cc.

Referenced by makewt().

5.42.3.2 void clTransformS::makewt (long, long ∗, float ∗) `[private]`

Definition at line 450 of file TransformS.cc.

References makeipt().

Referenced by cdft(), ddct(), ddst(), dfct(), dfst(), and rdft().

5.42.3.3 void clTransformS::makewt (long, long ∗, double ∗) [private]

Definition at line 3347 of file TransformS.cc.

References makeipt().

5.42.3.4 void clTransformS::makect (long, long ∗, float ∗) [private]

Definition at line 532 of file TransformS.cc.

Referenced by ddct(), ddst(), dfct(), dfst(), and rdft().

5.42.3.5 void clTransformS::makect (long, long ∗, double ∗) [private]

Definition at line 3429 of file TransformS.cc.

5.42.3.6 void clTransformS::cftfsub (long, float ∗, long ∗, long, float ∗) [private]

Definition at line 564 of file TransformS.cc.

References bitrv2(), bitrv208(), bitrv216(), cftf040(), cftf081(), cftf161(), cftf1st(), cftfx41(), cftleaf(), cftrec4(), and cftx020().

Referenced by cdft(), ddct(), ddst(), dfct(), dfst(), and rdft().

5.42.3.7 void clTransformS::cftfsub (long, double ∗, long ∗, long, double ∗) [private]

Definition at line 3461 of file TransformS.cc.

References bitrv2(), bitrv208(), bitrv216(), cftf040(), cftf081(), cftf161(), cftf1st(), cftfx41(), cftleaf(), cftrec4(), and cftx020().

5.42.3.8 void clTransformS::cftbsub (long, float ∗, long ∗, long, float ∗) [private]

Definition at line 597 of file TransformS.cc.

References bitrv208neg(), bitrv216neg(), bitrv2conj(), cftb040(), cftb1st(), cftf081(), cftf161(), cftfx41(), cftleaf(), cftrec4(), and cftx020().

Referenced by cdft(), ddct(), ddst(), and rdft().

5.42.3.9 void clTransformS::cftbsub (long, double ∗, long ∗, long, double ∗) [private]

Definition at line 3494 of file TransformS.cc.

References bitrv208neg(), bitrv216neg(), bitrv2conj(), cftb040(), cftb1st(), cftf081(), cftf161(), cftfx41(), cftleaf(), cftrec4(), and cftx020().

5.42.3.10 TS_INLINE void clTransformS::bitrv2 (long, long ∗, float ∗) [private]

Definition at line 630 of file TransformS.cc.

References TS_INLINE.

Referenced by cftfsub().

5.42.3.11 TS_INLINE void clTransformS::bitrv2 (long, long ∗, double ∗) [private]

Definition at line 3527 of file TransformS.cc.

References TS_INLINE.

5.42.3.12 TS_INLINE void clTransformS::bitrv2conj (long, long ∗, float ∗) [private]

Definition at line 977 of file TransformS.cc.

References TS_INLINE.

Referenced by cftbsub().

5.42.3.13 TS_INLINE void clTransformS::bitrv2conj (long, long ∗, double ∗) [private]

Definition at line 3874 of file TransformS.cc.

References TS_INLINE.

5.42.3.14 TS_INLINE void clTransformS::bitrv216 (float ∗) [private]

Definition at line 1332 of file TransformS.cc.

References TS_INLINE.

Referenced by cftfsub().

5.42.3.15 TS_INLINE void clTransformS::bitrv216 (double ∗) [private]

Definition at line 4229 of file TransformS.cc.

References TS_INLINE.

5.42.3.16 TS_INLINE void clTransformS::bitrv216neg (float ∗) [private]

Definition at line 1389 of file TransformS.cc.

References TS_INLINE.

Referenced by cftbsub().

5.42.3.17 TS_INLINE void clTransformS::bitrv216neg (double ∗) [private]

Definition at line 4286 of file TransformS.cc.

References TS_INLINE.

5.42.3.18 TS_INLINE void clTransformS::bitrv208 (float ∗) [private]

Definition at line 1459 of file TransformS.cc.

References TS_INLINE.

Referenced by cftfsub().

5.42.3.19 TS_INLINE void clTransformS::bitrv208 (double ∗) [private]

Definition at line 4356 of file TransformS.cc.

References TS_INLINE.

5.42.3.20 TS_INLINE void clTransformS::bitrv208neg (float ∗) [private]

Definition at line 1482 of file TransformS.cc.

References TS_INLINE.

Referenced by cftbsub().

5.42.3.21 TS_INLINE void clTransformS::bitrv208neg (double ∗) [private]

Definition at line 4379 of file TransformS.cc.

References TS_INLINE.

5.42.3.22 TS_INLINE void clTransformS::cftf1st (long, float ∗, float ∗) [private]

Definition at line 1518 of file TransformS.cc.

References TS_INLINE.

Referenced by cftfsub().

5.42.3.23 TS_INLINE void clTransformS::cftf1st (long, double ∗, double ∗) [private]

Definition at line 4415 of file TransformS.cc.

References TS_INLINE.

5.42.3.24 TS_INLINE void clTransformS::cftb1st (long, float ∗, float ∗) [private]

Definition at line 1724 of file TransformS.cc.

References TS_INLINE.

Referenced by cftbsub().

5.42.3.25 TS_INLINE void clTransformS::cftb1st (long, double ∗, double ∗) [private]

Definition at line 4621 of file TransformS.cc.

References TS_INLINE.

5.42.3.26 TS_INLINE void clTransformS::cftrec4 (long, float ∗, long, float ∗) [private]

Definition at line 1930 of file TransformS.cc.

References cftleaf(), cftmdl1(), cfttree(), and TS_INLINE.

Referenced by cftbsub(), and cftfsub().

5.42.3.27 TS_INLINE void clTransformS::cftrec4 (long, double ∗, long, double ∗) [private]

Definition at line 4827 of file TransformS.cc.

References cftleaf(), cftmdl1(), cfttree(), and TS_INLINE.

5.42.3.28 TS_INLINE long clTransformS::cfttree (long, long, long, float ∗, long, float ∗) [private]

Definition at line 1949 of file TransformS.cc.

References cftmdl1(), cftmdl2(), and TS_INLINE.

Referenced by cftrec4().

5.42.3.29 TS_INLINE long clTransformS::cfttree (long, long, long, double *, long, double *) [private]

Definition at line 4846 of file TransformS.cc.

References cftmdl1(), cftmdl2(), and TS_INLINE.

5.42.3.30 TS_INLINE void clTransformS::cftleaf (long, long, float *, long, float *) [private]

Definition at line 1983 of file TransformS.cc.

References cftf081(), cftf082(), cftf161(), cftf162(), cftmdl1(), cftmdl2(), and TS_-INLINE.

Referenced by cftbsub(), cftfsub(), and cftrec4().

5.42.3.31 TS_INLINE void clTransformS::cftleaf (long, long, double *, long, double *) [private]

Definition at line 4880 of file TransformS.cc.

References cftf081(), cftf082(), cftf161(), cftf162(), cftmdl1(), cftmdl2(), and TS_-INLINE.

5.42.3.32 TS_INLINE void clTransformS::cftmdl1 (long, float *, float *) [private]

Definition at line 2042 of file TransformS.cc.

References TS_INLINE.

Referenced by cftleaf(), cftrec4(), and cfttree().

5.42.3.33 TS_INLINE void clTransformS::cftmdl1 (long, double *, double *) [private]

Definition at line 4939 of file TransformS.cc.

References TS_INLINE.

5.42.3.34 TS_INLINE void clTransformS::cftmdl2 (long, float *, float *) [private]

Definition at line 2152 of file TransformS.cc.

References TS_INLINE.

Referenced by cftleaf(), and cfttree().

5.42.3.35 TS_INLINE void clTransformS::cftmdl2 (long, double ∗, double ∗) [private]

Definition at line 5049 of file TransformS.cc.

References TS_INLINE.

5.42.3.36 TS_INLINE void clTransformS::cftfx41 (long, float ∗, long, float ∗) [private]

Definition at line 2286 of file TransformS.cc.

References cftf081(), cftf082(), cftf161(), cftf162(), and TS_INLINE.

Referenced by cftbsub(), and cftfsub().

5.42.3.37 TS_INLINE void clTransformS::cftfx41 (long, double ∗, long, double ∗) [private]

Definition at line 5183 of file TransformS.cc.

References cftf081(), cftf082(), cftf161(), cftf162(), and TS_INLINE.

5.42.3.38 TS_INLINE void clTransformS::cftf161 (float ∗, float ∗) [private]

Definition at line 2302 of file TransformS.cc.

References TS_INLINE.

Referenced by cftbsub(), cftfsub(), cftfx41(), and cftleaf().

5.42.3.39 TS_INLINE void clTransformS::cftf161 (double ∗, double ∗) [private]

Definition at line 5199 of file TransformS.cc.

References TS_INLINE.

5.42.3.40 TS_INLINE void clTransformS::cftf162 (float ∗, float ∗) [private]

Definition at line 2461 of file TransformS.cc.

References TS_INLINE.

Referenced by cftfx41(), and cftleaf().

5.42.3.41 TS_INLINE void clTransformS::cftf162 (double ∗, double ∗) [private]

Definition at line 5358 of file TransformS.cc.

References TS_INLINE.

5.42.3.42 TS_INLINE void clTransformS::cftf081 (float ∗, float ∗) [private]

Definition at line 2644 of file TransformS.cc.

References TS_INLINE.

Referenced by cftbsub(), cftfsub(), cftfx41(), and cftleaf().

5.42.3.43 TS_INLINE void clTransformS::cftf081 (double ∗, double ∗) [private]

Definition at line 5541 of file TransformS.cc.

References TS_INLINE.

5.42.3.44 TS_INLINE void clTransformS::cftf082 (float ∗, float ∗) [private]

Definition at line 2706 of file TransformS.cc.

References TS_INLINE.

Referenced by cftfx41(), and cftleaf().

5.42.3.45 TS_INLINE void clTransformS::cftf082 (double ∗, double ∗) [private]

Definition at line 5603 of file TransformS.cc.

References TS_INLINE.

5.42.3.46 TS_INLINE void clTransformS::cftf040 (float ∗) [private]

Definition at line 2778 of file TransformS.cc.

References TS_INLINE.

Referenced by cftfsub().

5.42.3.47 TS_INLINE void clTransformS::cftf040 (double ∗) [private]

Definition at line 5675 of file TransformS.cc.

References TS_INLINE.

5.42.3.48 TS_INLINE void clTransformS::cftb040 (float *) [private]

Definition at line 2801 of file TransformS.cc.

References TS_INLINE.

Referenced by cftbsub().

5.42.3.49 TS_INLINE void clTransformS::cftb040 (double *) [private]

Definition at line 5698 of file TransformS.cc.

References TS_INLINE.

5.42.3.50 TS_INLINE void clTransformS::cftx020 (float *) [private]

Definition at line 2824 of file TransformS.cc.

References TS_INLINE.

Referenced by cftbsub(), and cftfsub().

5.42.3.51 TS_INLINE void clTransformS::cftx020 (double *) [private]

Definition at line 5721 of file TransformS.cc.

References TS_INLINE.

5.42.3.52 TS_INLINE void clTransformS::rftfsub (long, float *, long, float *) [private]

Definition at line 2837 of file TransformS.cc.

References TS_INLINE.

Referenced by ddct(), ddst(), dfct(), dfst(), and rdft().

5.42.3.53 TS_INLINE void clTransformS::rftfsub (long, double *, long, double *) [private]

Definition at line 5734 of file TransformS.cc.

References TS_INLINE.

5.42.3.54 TS_INLINE void clTransformS::rftbsub (long, float *, long, float *) [private]

Definition at line 2862 of file TransformS.cc.

References TS_INLINE.

Referenced by ddct(), ddst(), and rdft().

5.42.3.55 TS_INLINE void clTransformS::rftbsub (long, double ∗, long, double ∗) [private]

Definition at line 5759 of file TransformS.cc.

References TS_INLINE.

5.42.3.56 TS_INLINE void clTransformS::dctsub (long, float ∗, long, float ∗) [private]

Definition at line 2887 of file TransformS.cc.

References TS_INLINE.

Referenced by ddct(), and dfct().

5.42.3.57 TS_INLINE void clTransformS::dctsub (long, double ∗, long, double ∗) [private]

Definition at line 5784 of file TransformS.cc.

References TS_INLINE.

5.42.3.58 TS_INLINE void clTransformS::dstsub (long, float ∗, long, float ∗) [private]

Definition at line 2908 of file TransformS.cc.

References TS_INLINE.

Referenced by ddst(), and dfst().

5.42.3.59 TS_INLINE void clTransformS::dstsub (long, double ∗, long, double ∗) [private]

Definition at line 5805 of file TransformS.cc.

References TS_INLINE.

5.42.3.60 void clTransformS::cdft (long, long, float ∗, long ∗, float ∗)

Definition at line 127 of file TransformS.cc.

References cftbsub(), cftfsub(), and makewt().

Referenced by clDSPOp::FFTInitialize(), clDSPOp::FFTo(), and clDSPOp::IFFTo().

5.42.3.61 void clTransformS::cdft (long, long, double ∗, long ∗, double ∗)

Definition at line 3024 of file TransformS.cc.

References cftbsub(), cftfsub(), and makewt().

5.42.3.62 void clTransformS::rdft (long, long, float ∗, long ∗, float ∗)

Definition at line 144 of file TransformS.cc.

References cftbsub(), cftfsub(), makect(), makewt(), rftbsub(), and rftfsub().

Referenced by clDSPOp::FFTi(), clDSPOp::FFTInitialize(), clDSPOp::FFTo(), and clDSPOp::IFFTo().

5.42.3.63 void clTransformS::rdft (long, long, double ∗, long ∗, double ∗)

Definition at line 3041 of file TransformS.cc.

References cftbsub(), cftfsub(), makect(), makewt(), rftbsub(), and rftfsub().

5.42.3.64 void clTransformS::ddct (long, long, float ∗, long ∗, float ∗)

Definition at line 182 of file TransformS.cc.

References cftbsub(), cftfsub(), dctsub(), makect(), makewt(), rftbsub(), and rftfsub().

Referenced by clDSPOp::DCTi(), clDSPOp::DCTInitialize(), clDSPOp::DCTo(), clDSPOp::IDCTi(), and clDSPOp::IDCTo().

5.42.3.65 void clTransformS::ddct (long, long, double ∗, long ∗, double ∗)

Definition at line 3079 of file TransformS.cc.

References cftbsub(), cftfsub(), dctsub(), makect(), makewt(), rftbsub(), and rftfsub().

5.42.3.66 void clTransformS::ddst (long, long, float ∗, long ∗, float ∗)

Definition at line 231 of file TransformS.cc.

References cftbsub(), cftfsub(), dstsub(), makect(), makewt(), rftbsub(), and rftfsub().

5.42.3.67 void clTransformS::ddst (long, long, double ∗, long ∗, double ∗)

Definition at line 3128 of file TransformS.cc.

References cftbsub(), cftfsub(), dstsub(), makect(), makewt(), rftbsub(), and rftfsub().

5.42.3.68 void clTransformS::dfct (long, float $*$, float $*$, long $*$, float $*$)

Definition at line 280 of file TransformS.cc.

References cftfsub(), dctsub(), makect(), makewt(), and rftfsub().

5.42.3.69 void clTransformS::dfct (long, double $*$, double $*$, long $*$, double $*$)

Definition at line 3177 of file TransformS.cc.

References cftfsub(), dctsub(), makect(), makewt(), and rftfsub().

5.42.3.70 void clTransformS::dfst (long, float $*$, float $*$, long $*$, float $*$)

Definition at line 368 of file TransformS.cc.

References cftfsub(), dstsub(), makect(), makewt(), and rftfsub().

5.42.3.71 void clTransformS::dfst (long, double $*$, double $*$, long $*$, double $*$)

Definition at line 3265 of file TransformS.cc.

References cftfsub(), dstsub(), makect(), makewt(), and rftfsub().

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

- TransformS.hh
- TransformS.cc

5.43 clUserThreads Class Reference

Public Member Functions

- clUserThreads ()
- void ∗ Thread1 (void ∗)
- void ∗ Thread2 (void ∗)
- void ∗ Thread3 (void ∗)
- void Stop ()

Private Attributes

- volatile bool bRun

5.43.1 Constructor & Destructor Documentation

5.43.1.1 clUserThreads::clUserThreads () `[inline]`

Definition at line 11 of file Test.cc.

References bRun.

5.43.2 Member Function Documentation

5.43.2.1 void ∗ clUserThreads::Thread1 (void ∗)

Definition at line 48 of file Test.cc.

References bRun.

5.43.2.2 void ∗ clUserThreads::Thread2 (void ∗)

Definition at line 59 of file Test.cc.

References bRun.

5.43.2.3 void * clUserThreads::Thread3 (void *)

Definition at line 70 of file Test.cc.

References bRun.

5.43.2.4 void clUserThreads::Stop () [inline]

Definition at line 16 of file Test.cc.

References bRun.

Referenced by main().

5.43.3 Member Data Documentation

5.43.3.1 volatile bool clUserThreads::bRun [private]

Definition at line 9 of file Test.cc.

Referenced by clUserThreads(), Stop(), Thread1(), Thread2(), and Thread3().

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

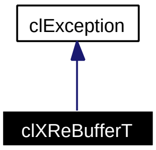
- Test.cc

5.44 clXReBufferT Class Reference

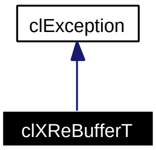
Exception class for clReBufferT.

`#include <ReBufferT.hh>`

Inheritance diagram for clXReBufferT:



Collaboration diagram for clXReBufferT:



Public Member Functions

- clXReBufferT (const char ∗cpErrorMsg)

5.44.1 Detailed Description

Exception class for clReBufferT.

Definition at line 40 of file ReBufferT.hh.

5.44.2 Constructor & Destructor Documentation

5.44.2.1 clXReBufferT::clXReBufferT (const char ∗ *cpErrorMsg*) `[inline]`

Definition at line 43 of file ReBufferT.hh.

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

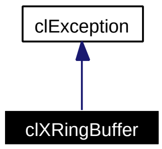
- ReBufferT.hh

5.45 clXRingBuffer Class Reference

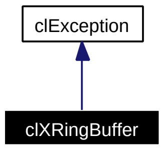
Exception class for clRingBuffer.

`#include <RingBuffer.hh>`

Inheritance diagram for clXRingBuffer:



Collaboration diagram for clXRingBuffer:



Public Member Functions

- clXRingBuffer (const char ∗cpErrorMsg)

5.45.1 Detailed Description

Exception class for clRingBuffer.

Definition at line 37 of file RingBuffer.hh.

5.45.2 Constructor & Destructor Documentation

5.45.2.1 clXRingBuffer::clXRingBuffer (const char ∗ *cpErrorMsg*) `[inline]`

Definition at line 40 of file RingBuffer.hh.

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

- RingBuffer.hh

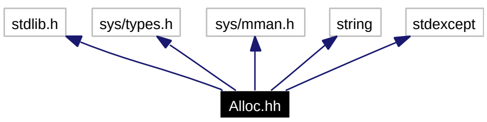
Chapter 6

libDSP File Documentation

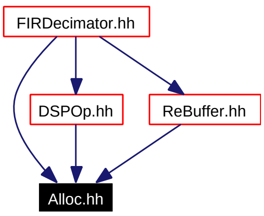
6.1 Alloc.hh File Reference

```
#include <stdlib.h>
#include <sys/types.h>
#include <sys/mman.h>
#include <string>
#include <stdexcept>
```

Include dependency graph for Alloc.hh:



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



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

Definition at line 36 of file Alloc.hh.

6.1.1.2 #define ALLOC_ALIGNMENT 32

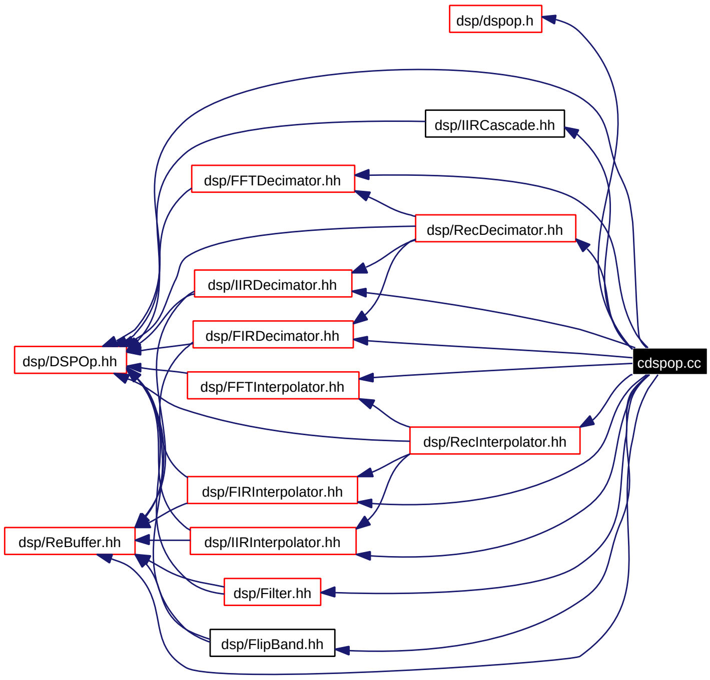
Definition at line 37 of file Alloc.hh.

Referenced by clAlloc::Resize(), and clAlloc::Size().

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/FlipBand.hh"
#include "dsp/ReBuffer.hh"
```

Include dependency graph for cdspop.cc:



Functions

- void dsp_init ()

  Initialize DSP library.

- signed long dsp_roundf (float fSrc)

  clDSPOp::Round

- signed long dsp_round (double dSrc)
- void dsp_addf (float ∗fpDest, float fSrc, long lCount)

  clDSPOp::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 l-Count)
- 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)

  clDSPOp::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)

  clDSPOp::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_cmulf (stpSCplx spDest, stSCplx sSrc, long lCount)
- void dsp_cmul (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_cmul2f (stpSCplx spDest, const stpSCplx spSrc, long lCount)
- void dsp_cmul2 (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_cmul3f (stpSCplx spDest, const stpSCplx spSrc1, const stpSCplx spSrc2, long lCount)
- void dsp_cmul3 (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 l-Count)
- void dsp_div3 (double ∗dpDest, const double ∗dpSrc1, const double ∗dpSrc2, long lCount)
- void dsp_cdiv3f (stpSCplx spDest, const stpSCplx spSrc1, const stpSCplx sp-Src2, long lCount)
- void dsp_cdiv3 (stpDCplx spDest, const stpDCplx spSrc1, const stpDCplx sp-Src2, 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, dou-ble 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)

  clDSPOp::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)

  clDSPOp::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)

  clDSPOp::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)

  clDSPOp::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)

  clDSPOp::Copy

- void dsp_copy (double ∗dpDest, const double ∗dpSrc, long lCount)
- void dsp_ccopyf (stpSCplx spDest, const stpSCplx spSrc, long lCount)
- void dsp_ccopy (stpDCplx spDest, const stpDCplx spSrc, long lCount)

- float dsp_convolvef (const float *fpSrc1, const float *fpSrc2, long lCount)

  clDSPOp::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 *dp-Src2, 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 *dp-Src2, 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 l-Count)
- void dsp_minmaxf (float *fpMin, float *fpMax, const float *fpSrc, long l-Count)

  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 l-Count)

    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 l-Count)
- 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 sp-Cart, 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 *fp-Phase, 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_crosscorrf (const float *fpSrc1, const float *fpSrc2, long lCount)

  clDSPOp::CrossCorr

- double dsp_crosscorr (const double *dpSrc1, const double *dpSrc2, long l-Count)
- 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 l-Delay, 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 *dp-Src2, 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 sp-Src, long lCount)
- void dsp_decimatef (float *fpDest, const float *fpSrc, long lFactor, long l-Count)

  clDSPOp::Decimate

- void dsp_decimate (double *dpDest, const double *dpSrc, long lFactor, long l-Count)
- void dsp_decimateavgf (float *fpDest, const float *fpSrc, long lFactor, long l-Count)

  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 l-Count)

  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 l-SrcCount)

  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 *dp-Src, 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_bartlettf (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_harrisf (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 l-Count)

  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_chebyshevf (dsp_t dspInst, float ∗fpDest, float fGamma, long lCount)

  clDSPOp::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)

  clDSPOp::ReBuffer

- long dsp_rebuffer (dsp_t dspInst, double ∗dpDest, const double ∗dpSrc, long lDestCount, long lSrcCount)
- float dsp_deg2radf (dsp_t dspInst, float fSource)

  clDSPOp::DegToRad

- double dsp_deg2rad (dsp_t dspInst, double dSource)
- float dsp_rad2degf (dsp_t dspInst, float fSource)

  clDSPOp::RadToDeg

- double dsp_rad2deg (dsp_t dspInst, double dSource)
- void dsp_fir_allocatef (dsp_t dspInst, const float ∗fpCoeff, long lCount)

  clDSPOp::FIRAllocate

- void dsp_fir_allocate (dsp_t dspInst, const double ∗dpCoeff, long lCount)
- void dsp_fir_free (dsp_t dspInst)

  clDSPOp::FIRFree

- void dsp_fir_filterf (dsp_t dspInst, float ∗fpVect, long lCount)

  clDSPOp::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)

  clDSPOp::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)

  clDSPOp::IIRInitialize

- void dsp_iir_init (dsp_t dspInst, const double ∗dpCoeffs)
- void dsp_iir_filterf (dsp_t dspInst, float ∗fpVect, long lCount)

  clDSPOp::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_cfftf_nip (dsp_t dspInst, stpSCplx spDest, const stpSCplx spSrc)
- void dsp_cfft_nip (dsp_t dspInst, stpDCplx spDest, const stpDCplx spSrc)
- void dsp_ifftf_nip (dsp_t dspInst, float *fpDest, const stpSCplx spSrc)

  clDSPOp::IFFTo

- void dsp_ifft_nip (dsp_t dspInst, double *dpDest, const stpDCplx spSrc)
- void dsp_cifftf_nip (dsp_t dspInst, stpSCplx spDest, const stpSCplx spSrc)
- void dsp_cifft_nip (dsp_t dspInst, stpDCplx spDest, const stpDCplx spSrc)
- void dsp_dct_init (dsp_t dspInst, long lSize)

  clDSPOp::DCTInitialize

- void dsp_dct_uninit (dsp_t dspInst)

  clDSPOp::DCTUninitialize

- void dsp_dctf (dsp_t dspInst, float *fpVect)

  clDSPOp::DCTi

- void dsp_dct (dsp_t dspInst, double *dpVect)
- void dsp_dctf_nip (dsp_t dspInst, float *fpDest, const float *fpSrc)

  clDSPOp::DCTo

- void dsp_dct_nip (dsp_t dspInst, double *dpDest, const double *dpSrc)
- void dsp_idctf (dsp_t dspInst, float *fpVect)

  clDSPOp::IDCTi

- void dsp_idct (dsp_t dspInst, double *dpVect)
- void dsp_idctf_nip (dsp_t dspInst, float *fpDest, const float *fpSrc)

  clDSPOp::IDCTo

- void dsp_idct_nip (dsp_t dspInst, double *dpDest, const double *dpSrc)
- 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 l-Count)
- 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_coeffsf (dsp_filter_t dspfilterInst, const float ∗fpFiltCoeffs)

  clFilter::SetCoeffs

- void dsp_filter_set_coeffs (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_coeffsf (dsp_filter_t dspfilterInst, float ∗fpFiltCoeffs)

  clFilter::GetCoeffs

- void dsp_filter_get_coeffs (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 fSampleRate, 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_flip_t dsp_flip_new ()

  Creates new instance of band flipping object.

- void dsp_flip_delete (dsp_flip_t dspflipInst)

  Deletes instance of band flipping object.

- void dsp_flip_initf (dsp_flip_t dspflipInst, long lSize)

  clFlipBand::Initialize

- void dsp_flip_init (dsp_flip_t dspflipInst, long lSize)
- void dsp_flip_uninit (dsp_flip_t dspflipInst)

  clFlipBand::Uninitialize

- void dsp_flip_putf (dsp_flip_t dspflipInst, const float ∗fpSrc, long lCount)

  clFlipBand::Put

- void dsp_flip_put (dsp_flip_t dspflipInst, const double ∗dpSrc, long lCount)
- void dsp_flip_getf (dsp_flip_t dspflipInst, float ∗fpDest, long lCount)

  clFlipBand::Get

- void dsp_flip_get (dsp_flip_t dspflipInst, double ∗dpDest, long lCount)
- void dsp_flip_clear (dsp_flip_t dspflipInst)

  clFlipBand::Clear

- 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 dsprebufCopy-Src)

  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.

Definition at line 53 of file cdspop.cc.

6.2.1.2 signed long dsp_roundf (float *fSrc*)

clDSPOp::Round

Definition at line 62 of file cdspop.cc.

References clDSPOp::Round().

6.2.1.3 signed long dsp_round (double *dSrc*)

Definition at line 68 of file cdspop.cc.

References clDSPOp::Round().

6.2.1.4 void dsp_addf (float $*$ *fpDest*, float *fSrc*, long *lCount*)

clDSPOp::Add

Definition at line 74 of file cdspop.cc.

References clDSPOp::Add().

6.2.1.5 void dsp_add (double $*$ *dpDest*, double *dSrc*, long *lCount*)

Definition at line 80 of file cdspop.cc.

References clDSPOp::Add().

6.2.1.6 void dsp_caddf (stpSCplx *spDest*, stSCplx *sSrc*, long *lCount*)

Definition at line 86 of file cdspop.cc.

References clDSPOp::Add(), stpSCplx, and stSCplx.

6.2.1.7 void dsp_cadd (stpDCplx *spDest*, stDCplx *sSrc*, long *lCount*)

Definition at line 92 of file cdspop.cc.

References clDSPOp::Add(), stDCplx, and stpDCplx.

6.2.1.8 void dsp_add2f (float ∗ *fpDest*, const float ∗ *fpSrc*, long *lCount*)

Definition at line 98 of file cdspop.cc.

References clDSPOp::Add().

6.2.1.9 void dsp_add2 (double ∗ *dpDest*, const double ∗ *dpSrc*, long *lCount*)

Definition at line 104 of file cdspop.cc.

References clDSPOp::Add().

6.2.1.10 void dsp_cadd2f (stpSCplx *spDest*, const stpSCplx *spSrc*, long *lCount*)

Definition at line 110 of file cdspop.cc.

References clDSPOp::Add(), and stpSCplx.

6.2.1.11 void dsp_cadd2 (stpDCplx *spDest*, const stpDCplx *spSrc*, long *lCount*)

Definition at line 116 of file cdspop.cc.

References clDSPOp::Add(), and stpDCplx.

6.2.1.12 void dsp_add3f (float ∗ *fpDest*, const float ∗ *fpSrc1*, const float ∗ *fpSrc2*, long *lCount*)

Definition at line 122 of file cdspop.cc.

References clDSPOp::Add().

6.2.1.13 void dsp_add3 (double ∗ *dpDest*, const double ∗ *dpSrc1*, const double ∗ *dpSrc2*, long *lCount*)

Definition at line 129 of file cdspop.cc.

References clDSPOp::Add().

6.2.1.14 void dsp_cadd3f (stpSCplx *spDest*, const stpSCplx *spSrc1*, const stpSCplx *spSrc2*, long *lCount*)

Definition at line 136 of file cdspop.cc.

References clDSPOp::Add(), and stpSCplx.

6.2.1.15 void dsp_cadd3 (stpDCplx *spDest*, const stpDCplx *spSrc1*, const stpDCplx *spSrc2*, long *lCount*)

Definition at line 143 of file cdspop.cc.

References clDSPOp::Add(), and stpDCplx.

6.2.1.16 void dsp_subf (float * *fpDest*, float *fSrc*, long *lCount*)

clDSPOp::Sub

Definition at line 150 of file cdspop.cc.

References clDSPOp::Sub().

6.2.1.17 void dsp_sub (double * *dpDest*, double *dSrc*, long *lCount*)

Definition at line 156 of file cdspop.cc.

References clDSPOp::Sub().

6.2.1.18 void dsp_csubf (stpSCplx *spDest*, stSCplx *sSrc*, long *lCount*)

Definition at line 162 of file cdspop.cc.

References stpSCplx, stSCplx, and clDSPOp::Sub().

6.2.1.19 void dsp_csub (stpDCplx *spDest*, stDCplx *sSrc*, long *lCount*)

Definition at line 168 of file cdspop.cc.

References stDCplx, stpDCplx, and clDSPOp::Sub().

6.2.1.20 void dsp_sub2f (float * *fpDest*, const float * *fpSrc*, long *lCount*)

Definition at line 174 of file cdspop.cc.

References clDSPOp::Sub().

6.2.1.21 void dsp_sub2 (double * *dpDest*, const double * *dpSrc*, long *lCount*)

Definition at line 180 of file cdspop.cc.

References clDSPOp::Sub().

6.2.1.22 void dsp_csub2f (stpSCplx *spDest*, const stpSCplx *spSrc*, long *lCount*)

Definition at line 186 of file cdspop.cc.

References stpSCplx, and clDSPOp::Sub().

6.2.1.23 void dsp_csub2 (stpDCplx *spDest*, const stpDCplx *spSrc*, long *lCount*)

Definition at line 192 of file cdspop.cc.

References stpDCplx, and clDSPOp::Sub().

6.2.1.24 void dsp_sub3f (float ∗ *fpDest*, const float ∗ *fpSrc1*, const float ∗ *fpSrc2*, long *lCount*)

Definition at line 198 of file cdspop.cc.

References clDSPOp::Sub().

6.2.1.25 void dsp_sub3 (double ∗ *dpDest*, const double ∗ *dpSrc1*, const double ∗ *dpSrc2*, long *lCount*)

Definition at line 205 of file cdspop.cc.

References clDSPOp::Sub().

6.2.1.26 void dsp_csub3f (stpSCplx *spDest*, const stpSCplx *spSrc1*, const stpSCplx *spSrc2*, long *lCount*)

Definition at line 212 of file cdspop.cc.

References stpSCplx, and clDSPOp::Sub().

6.2.1.27 void dsp_csub3 (stpDCplx *spDest*, const stpDCplx *spSrc1*, const stpDCplx *spSrc2*, long *lCount*)

Definition at line 219 of file cdspop.cc.

References stpDCplx, and clDSPOp::Sub().

6.2.1.28 void dsp_mulf (float ∗ *fpDest*, float *fSrc*, long *lCount*)

clDSPOp::Mul

Definition at line 226 of file cdspop.cc.

References clDSPOp::Mul().

6.2.1.29 void dsp_mul (double * *dpDest*, double *dSrc*, long *lCount*)

Definition at line 232 of file cdspop.cc.

References clDSPOp::Mul().

6.2.1.30 void dsp_chmulf (stpSCplx *spDest*, float *fSrc*, long *lCount*)

Definition at line 238 of file cdspop.cc.

References clDSPOp::Mul(), and stpSCplx.

6.2.1.31 void dsp_chmul (stpDCplx *spDest*, double *dSrc*, long *lCount*)

Definition at line 244 of file cdspop.cc.

References clDSPOp::Mul(), and stpDCplx.

6.2.1.32 void dsp_cmulf (stpSCplx *spDest*, stSCplx *sSrc*, long *lCount*)

Definition at line 250 of file cdspop.cc.

References clDSPOp::Mul(), stpSCplx, and stSCplx.

6.2.1.33 void dsp_cmul (stpDCplx *spDest*, stDCplx *sSrc*, long *lCount*)

Definition at line 256 of file cdspop.cc.

References clDSPOp::Mul(), stDCplx, and stpDCplx.

6.2.1.34 void dsp_mulf_nip (float * *fpDest*, const float * *fpSrc1*, float *fSrc2*, long *lCount*)

Definition at line 262 of file cdspop.cc.

References clDSPOp::Mul().

6.2.1.35 void dsp_mul_nip (double * *dpDest*, const double * *dpSrc1*, double *dSrc2*, long *lCount*)

Definition at line 269 of file cdspop.cc.

References clDSPOp::Mul().

6.2.1.36 void dsp_mul2f (float * *fpDest*, const float * *fpSrc*, long *lCount*)

Definition at line 276 of file cdspop.cc.

References clDSPOp::Mul().

6.2.1.37 void dsp_mul2 (double * *dpDest*, const double * *dpSrc*, long *lCount*)

Definition at line 282 of file cdspop.cc.

References clDSPOp::Mul().

6.2.1.38 void dsp_chmul2f (stpSCplx *spDest*, const float * *fpSrc*, long *lCount*)

Definition at line 288 of file cdspop.cc.

References clDSPOp::Mul(), and stpSCplx.

6.2.1.39 void dsp_chmul2 (stpDCplx *spDest*, const double * *dpSrc*, long *lCount*)

Definition at line 294 of file cdspop.cc.

References clDSPOp::Mul(), and stpDCplx.

6.2.1.40 void dsp_cmul2f (stpSCplx *spDest*, const stpSCplx *spSrc*, long *lCount*)

Definition at line 300 of file cdspop.cc.

References clDSPOp::Mul(), and stpSCplx.

6.2.1.41 void dsp_cmul2 (stpDCplx *spDest*, const stpDCplx *spSrc*, long *lCount*)

Definition at line 306 of file cdspop.cc.

References clDSPOp::Mul(), and stpDCplx.

6.2.1.42 void dsp_mul3f (float * *fpDest*, const float * *fpSrc1*, const float * *fpSrc2*, long *lCount*)

Definition at line 312 of file cdspop.cc.

References clDSPOp::Mul().

6.2.1.43 void dsp_mul3 (double * *dpDest*, const double * *dpSrc1*, const double * *dpSrc2*, long *lCount*)

Definition at line 319 of file cdspop.cc.

References clDSPOp::Mul().

6.2.1.44 void dsp_cmul3f (stpSCplx *spDest*, const stpSCplx *spSrc1*, const stpSCplx *spSrc2*, long *lCount*)

Definition at line 326 of file cdspop.cc.

References clDSPOp::Mul(), and stpSCplx.

6.2.1.45 void dsp_cmul3 (stpDCplx *spDest*, const stpDCplx *spSrc1*, const stpDCplx *spSrc2*, long *lCount*)

Definition at line 333 of file cdspop.cc.

References clDSPOp::Mul(), and stpDCplx.

6.2.1.46 void dsp_ccmulf (stpSCplx *spDest*, const stpSCplx *spSrc*, long *lCount*)

Definition at line 340 of file cdspop.cc.

References clDSPOp::MulC(), and stpSCplx.

6.2.1.47 void dsp_ccmul (stpDCplx *spDest*, const stpDCplx *spSrc*, long *lCount*)

Definition at line 346 of file cdspop.cc.

References clDSPOp::MulC(), and stpDCplx.

6.2.1.48 void dsp_ccmulf_nip (stpSCplx *spDest*, const stpSCplx *spSrc1*, const stpSCplx *spSrc2*, long *lCount*)

Definition at line 352 of file cdspop.cc.

References clDSPOp::MulC(), and stpSCplx.

6.2.1.49 void dsp_ccmul_nip (stpDCplx *spDest*, const stpDCplx *spSrc1*, const stpDCplx *spSrc2*, long *lCount*)

Definition at line 359 of file cdspop.cc.

References clDSPOp::MulC(), and stpDCplx.

6.2.1.50 void dsp_divf (float * *fpDest*, float *fSrc*, long *lCount*)

clDSPOp::Div

Definition at line 366 of file cdspop.cc.

References clDSPOp::Div().

6.2.1.51 void dsp_div (double * *dpDest*, double *dSrc*, long *lCount*)

Definition at line 372 of file cdspop.cc.

References clDSPOp::Div().

6.2.1.52 void dsp_cdivf (stpSCplx *spDest*, stSCplx *sSrc*, long *lCount*)

Definition at line 378 of file cdspop.cc.

References clDSPOp::Div(), stpSCplx, and stSCplx.

6.2.1.53 void dsp_cdiv (stpDCplx *spDest*, stDCplx *sSrc*, long *lCount*)

Definition at line 384 of file cdspop.cc.

References clDSPOp::Div(), stDCplx, and stpDCplx.

6.2.1.54 void dsp_div2f (float * *fpDest*, const float * *fpSrc*, long *lCount*)

Definition at line 390 of file cdspop.cc.

References clDSPOp::Div().

6.2.1.55 void dsp_div2 (double * *dpDest*, const double * *dpSrc*, long *lCount*)

Definition at line 396 of file cdspop.cc.

References clDSPOp::Div().

6.2.1.56 void dsp_cdiv2f (stpSCplx *spDest*, const stpSCplx *spSrc*, long *lCount*)

Definition at line 402 of file cdspop.cc.

References clDSPOp::Div(), and stpSCplx.

6.2.1.57 void dsp_cdiv2 (stpDCplx *spDest*, const stpDCplx *spSrc*, long *lCount*)

Definition at line 408 of file cdspop.cc.

References clDSPOp::Div(), and stpDCplx.

6.2.1.58 void dsp_div3f (float * *fpDest*, const float * *fpSrc1*, const float * *fpSrc2*, long *lCount*)

Definition at line 414 of file cdspop.cc.

References clDSPOp::Div().

6.2.1.59 void dsp_div3 (double * *dpDest*, const double * *dpSrc1*, const double * *dpSrc2*, long *lCount*)

Definition at line 421 of file cdspop.cc.

References clDSPOp::Div().

6.2.1.60 void dsp_cdiv3f (stpSCplx *spDest*, const stpSCplx *spSrc1*, const stpSCplx *spSrc2*, long *lCount*)

Definition at line 428 of file cdspop.cc.

References clDSPOp::Div(), and stpSCplx.

6.2.1.61 void dsp_cdiv3 (stpDCplx *spDest*, const stpDCplx *spSrc1*, const stpDCplx *spSrc2*, long *lCount*)

Definition at line 435 of file cdspop.cc.

References clDSPOp::Div(), and stpDCplx.

6.2.1.62 void dsp_div1xf (float $*$ *fpVect*, long *lCount*)

clDSPOp::Div1x

Definition at line 442 of file cdspop.cc.

References clDSPOp::Div1x().

6.2.1.63 void dsp_div1x (double $*$ *dpVect*, long *lCount*)

Definition at line 448 of file cdspop.cc.

References clDSPOp::Div1x().

6.2.1.64 void dsp_div1xf_nip (float $*$ *fpDest*, const float $*$ *fpSrc*, long *lCount*)

Definition at line 454 of file cdspop.cc.

References clDSPOp::Div1x().

6.2.1.65 void dsp_div1x_nip (double $*$ *dpDest*, const double $*$ *dpSrc*, long *lCount*)

Definition at line 460 of file cdspop.cc.

References clDSPOp::Div1x().

6.2.1.66 void dsp_muladdf (float $*$ *fpVect*, float *fMul*, float *fAdd*, long *lCount*)

clDSPOp::MulAdd

Definition at line 466 of file cdspop.cc.

References clDSPOp::MulAdd().

6.2.1.67 void dsp_muladd (double ∗ *dpVect*, double *dMul*, double *dAdd*, long *lCount*)

Definition at line 472 of file cdspop.cc.

References clDSPOp::MulAdd().

6.2.1.68 void dsp_muladdf_nip (float ∗ *fpDest*, const float ∗ *fpSrc*, float *fMul*, float *fAdd*, long *lCount*)

Definition at line 478 of file cdspop.cc.

References clDSPOp::MulAdd().

6.2.1.69 void dsp_muladd_nip (double ∗ *dpDest*, const double ∗ *dpSrc*, double *dMul*, double *dAdd*, long *lCount*)

Definition at line 485 of file cdspop.cc.

References clDSPOp::MulAdd().

6.2.1.70 void dsp_absf (float ∗ *fpVect*, long *lCount*)

clDSPOp::Abs

Definition at line 492 of file cdspop.cc.

References clDSPOp::Abs().

6.2.1.71 void dsp_abs (double ∗ *dpVect*, long *lCount*)

Definition at line 498 of file cdspop.cc.

References clDSPOp::Abs().

6.2.1.72 void dsp_absf_nip (float ∗ *fpDest*, const float ∗ *fpSrc*, long *lCount*)

Definition at line 504 of file cdspop.cc.

References clDSPOp::Abs().

6.2.1.73 void dsp_abs_nip (double ∗ *dpDest*, const double ∗ *dpSrc*, long *lCount*)

Definition at line 510 of file cdspop.cc.

References clDSPOp::Abs().

6.2.1.74 void dsp_sqrtf (float * *fpVect*, long *lCount*)

clDSPOp::Sqrt

Definition at line 516 of file cdspop.cc.

References clDSPOp::Sqrt().

6.2.1.75 void dsp_sqrt (double * *dpVect*, long *lCount*)

Definition at line 522 of file cdspop.cc.

References clDSPOp::Sqrt().

6.2.1.76 void dsp_sqrtf_nip (float * *fpDest*, const float * *fpSrc*, long *lCount*)

Definition at line 528 of file cdspop.cc.

References clDSPOp::Sqrt().

6.2.1.77 void dsp_sqrt_nip (double * *dpDest*, const double * *dpSrc*, long *lCount*)

Definition at line 534 of file cdspop.cc.

References clDSPOp::Sqrt().

6.2.1.78 void dsp_zerof (float * *fpVect*, long *lCount*)

clDSPOp::Zero

Definition at line 540 of file cdspop.cc.

References clDSPOp::Zero().

6.2.1.79 void dsp_zero (double * *dpVect*, long *lCount*)

Definition at line 546 of file cdspop.cc.

References clDSPOp::Zero().

6.2.1.80 void dsp_czerof (stpSCplx *spVect*, long *lCount*)

Definition at line 552 of file cdspop.cc.

References stpSCplx, and clDSPOp::Zero().

6.2.1.81 void dsp_czero (stpDCplx *spVect*, long *lCount*)

Definition at line 558 of file cdspop.cc.

References stpDCplx, and clDSPOp::Zero().

6.2.1.82 void dsp_setf (float ∗ *fpDest*, float *fSrc*, long *lCount*)

clDSPOp::Set

Definition at line 564 of file cdspop.cc.

References clDSPOp::Set().

6.2.1.83 void dsp_set (double ∗ *dpDest*, double *dSrc*, long *lCount*)

Definition at line 570 of file cdspop.cc.

References clDSPOp::Set().

6.2.1.84 void dsp_csetf (stpSCplx *spDest*, stSCplx *sSrc*, long *lCount*)

Definition at line 576 of file cdspop.cc.

References clDSPOp::Set(), stpSCplx, and stSCplx.

6.2.1.85 void dsp_cset (stpDCplx *spDest*, stDCplx *sSrc*, long *lCount*)

Definition at line 582 of file cdspop.cc.

References clDSPOp::Set(), stDCplx, and stpDCplx.

6.2.1.86 void dsp_set2f (float ∗ *fpDest*, float *fSrc*, long *lStart*, long *lCount*, long *lLength*)

Definition at line 588 of file cdspop.cc.

References clDSPOp::Set().

6.2.1.87 void dsp_set2 (double ∗ *dpDest*, double *dSrc*, long *lStart*, long *lCount*, long *lLength*)

Definition at line 595 of file cdspop.cc.

References clDSPOp::Set().

6.2.1.88 void dsp_cset2f (stpSCplx *spDest*, stSCplx *sSrc*, long *lStart*, long *lCount*, long *lLength*)

Definition at line 602 of file cdspop.cc.

References clDSPOp::Set(), stpSCplx, and stSCplx.

6.2.1.89 void dsp_cset2 (stpDCplx *spDest*, stDCplx *sSrc*, long *lStart*, long *lCount*, long *lLength*)

Definition at line 609 of file cdspop.cc.

References clDSPOp::Set(), stDCplx, and stpDCplx.

6.2.1.90 void dsp_clipf (float ∗ *fpVect*, float *fValue*, long *lCount*)

clDSPOp::Clip

Definition at line 616 of file cdspop.cc.

References clDSPOp::Clip().

6.2.1.91 void dsp_clip (double ∗ *dpVect*, double *dValue*, long *lCount*)

Definition at line 622 of file cdspop.cc.

References clDSPOp::Clip().

6.2.1.92 void dsp_clipf_nip (float ∗ *fpDest*, const float ∗ *fpSrc*, float *fValue*, long *lCount*)

Definition at line 628 of file cdspop.cc.

References clDSPOp::Clip().

6.2.1.93 void dsp_clip_nip (double ∗ *dpDest*, const double ∗ *dpSrc*, double *dValue*, long *lCount*)

Definition at line 635 of file cdspop.cc.

References clDSPOp::Clip().

6.2.1.94 void dsp_clip2f (float ∗ *fpVect*, float *fMin*, float *fMax*, long *lCount*)

Definition at line 642 of file cdspop.cc.

References clDSPOp::Clip().

6.2.1.95 void dsp_clip2 (double ∗ *dpVect*, double *dMin*, double *dMax*, long *lCount*)

Definition at line 648 of file cdspop.cc.

References clDSPOp::Clip().

6.2.1.96 void dsp_clip2f_nip (float ∗ *fpDest*, const float ∗ *fpSrc*, float *fMin*, float *fMax*, long *lCount*)

Definition at line 654 of file cdspop.cc.

References clDSPOp::Clip().

6.2.1.97 void dsp_clip2_nip (double ∗ *dpDest*, const double ∗ *dpSrc*, double *dMin*, double *dMax*, long *lCount*)

Definition at line 661 of file cdspop.cc.

References clDSPOp::Clip().

6.2.1.98 void dsp_clipzerof (float ∗ *fpVect*, long *lCount*)

clDSPOp::ClipZero

Definition at line 668 of file cdspop.cc.

References clDSPOp::ClipZero().

6.2.1.99 void dsp_clipzero (double ∗ *dpVect*, long *lCount*)

Definition at line 674 of file cdspop.cc.

References clDSPOp::ClipZero().

6.2.1.100 void dsp_clipzerof_nip (float ∗ *fpDest*, const float ∗ *fpSrc*, long *lCount*)

Definition at line 680 of file cdspop.cc.

References clDSPOp::ClipZero().

6.2.1.101 void dsp_clipzero_nip (double ∗ *dpDest*, const double ∗ *dpSrc*, long *lCount*)

Definition at line 686 of file cdspop.cc.

References clDSPOp::ClipZero().

6.2.1.102 void dsp_copyf (float ∗ *fpDest*, const float ∗ *fpSrc*, long *lCount*)

clDSPOp::Copy

Definition at line 692 of file cdspop.cc.

References clDSPOp::Copy().

6.2.1.103 void dsp_copy (double ∗ *dpDest*, const double ∗ *dpSrc*, long *lCount*)

Definition at line 698 of file cdspop.cc.

References clDSPOp::Copy().

6.2.1.104 void dsp_ccopyf (stpSCplx *spDest*, const stpSCplx *spSrc*, long *lCount*)

Definition at line 704 of file cdspop.cc.

References clDSPOp::Copy(), and stpSCplx.

6.2.1.105 void dsp_ccopy (stpDCplx *spDest*, const stpDCplx *spSrc*, long *lCount*)

Definition at line 710 of file cdspop.cc.

References clDSPOp::Copy(), and stpDCplx.

6.2.1.106 float dsp_convolvef (const float ∗ *fpSrc1*, const float ∗ *fpSrc2*, long *lCount*)

clDSPOp::Convolve

Definition at line 716 of file cdspop.cc.

References clDSPOp::Convolve().

6.2.1.107 double dsp_convolve (const double ∗ *dpSrc1*, const double ∗ *dpSrc2*, long *lCount*)

Definition at line 722 of file cdspop.cc.

References clDSPOp::Convolve().

6.2.1.108 void dsp_convolve2f (float ∗ *fpDest*, const float ∗ *fpSrc1*, const float ∗ *fpSrc2*, long *lCount*)

Definition at line 728 of file cdspop.cc.

References clDSPOp::Convolve().

6.2.1.109 void dsp_convolve2 (double ∗ *dpDest*, const double ∗ *dpSrc1*, const double ∗ *dpSrc2*, long *lCount*)

Definition at line 735 of file cdspop.cc.

References clDSPOp::Convolve().

6.2.1.110 float dsp_correlatef (const float * *fpSrc1*, const float * *fpSrc2*, long *lCount*)

clDSPOp::Correlate

Definition at line 742 of file cdspop.cc.

References clDSPOp::Correlate().

6.2.1.111 double dsp_correlate (const double * *dpSrc1*, const double * *dpSrc2*, long *lCount*)

Definition at line 748 of file cdspop.cc.

References clDSPOp::Correlate().

6.2.1.112 void dsp_correlate2f (float * *fpDest*, const float * *fpSrc1*, const float * *fpSrc2*, long *lCount*)

Definition at line 754 of file cdspop.cc.

References clDSPOp::Correlate().

6.2.1.113 void dsp_correlate2 (double * *dpDest*, const double * *dpSrc1*, const double * *dpSrc2*, long *lCount*)

Definition at line 761 of file cdspop.cc.

References clDSPOp::Correlate().

6.2.1.114 float dsp_autocorrf (const float * *fpSrc*, long *lCount*)

clDSPOp::AutoCorrelate

Definition at line 768 of file cdspop.cc.

References clDSPOp::AutoCorrelate().

6.2.1.115 double dsp_autocorr (const double * *dpSrc*, long *lCount*)

Definition at line 774 of file cdspop.cc.

References clDSPOp::AutoCorrelate().

6.2.1.116 void dsp_autocorr2f (float * *fpDest*, const float * *fpSrc*, long *lCount*)

Definition at line 780 of file cdspop.cc.

References clDSPOp::AutoCorrelate().

6.2.1.117 void dsp_autocorr2 (double * *dpDest*, const double * *dpSrc*, long *lCount*)

Definition at line 786 of file cdspop.cc.

References clDSPOp::AutoCorrelate().

6.2.1.118 float dsp_dotproductf (const float * *fpSrc1*, const float * *fpSrc2*, long *lCount*)

clDSPOp::DotProduct

Definition at line 792 of file cdspop.cc.

References clDSPOp::DotProduct().

6.2.1.119 double dsp_dotproduct (const double * *dpSrc1*, const double * *dpSrc2*, long *lCount*)

Definition at line 798 of file cdspop.cc.

References clDSPOp::DotProduct().

6.2.1.120 void dsp_minmaxf (float * *fpMin*, float * *fpMax*, const float * *fpSrc*, long *lCount*)

clDSPOp::MinMax

Definition at line 804 of file cdspop.cc.

References clDSPOp::MinMax().

6.2.1.121 void dsp_minmax (double * *dpMin*, double * *dpMax*, const double * *dpSrc*, long *lCount*)

Definition at line 810 of file cdspop.cc.

References clDSPOp::MinMax().

6.2.1.122 float dsp_meanf (const float * *fpSrc*, long *lCount*)

clDSPOp::Mean

Definition at line 816 of file cdspop.cc.

References clDSPOp::Mean().

6.2.1.123 double dsp_mean (const double * *dpSrc*, long *lCount*)

Definition at line 822 of file cdspop.cc.

References clDSPOp::Mean().

6.2.1.124 float dsp_medianf (const float ∗ *fpSrc*, long *lCount*)

Definition at line 828 of file cdspop.cc.

References clDSPOp::Median().

6.2.1.125 double dsp_median (const double ∗ *dpSrc*, long *lCount*)

Definition at line 834 of file cdspop.cc.

References clDSPOp::Median().

6.2.1.126 void dsp_negatef (float ∗ *fpVect*, long *lCount*)

clDSPOp::Negate

Definition at line 840 of file cdspop.cc.

References clDSPOp::Negate().

6.2.1.127 void dsp_negate (double ∗ *dpVect*, long *lCount*)

Definition at line 846 of file cdspop.cc.

References clDSPOp::Negate().

6.2.1.128 void dsp_negatef_nip (float ∗ *fpDest*, const float ∗ *fpSrc*, long *lCount*)

Definition at line 852 of file cdspop.cc.

References clDSPOp::Negate().

6.2.1.129 void dsp_negate_nip (double ∗ *dpDest*, const double ∗ *dpSrc*, long *lCount*)

Definition at line 858 of file cdspop.cc.

References clDSPOp::Negate().

6.2.1.130 void dsp_normalizef (float ∗ *fpVect*, long *lCount*)

clDSPOp::Normalize

Definition at line 864 of file cdspop.cc.

References clDSPOp::Normalize().

6.2.1.131 void dsp_normalize (double * *dpVect*, long *lCount*)

Definition at line 870 of file cdspop.cc.

References clDSPOp::Normalize().

6.2.1.132 void dsp_normalizef_nip (float * *fpDest*, const float * *fpSrc*, long *lCount*)

Definition at line 876 of file cdspop.cc.

References clDSPOp::Normalize().

6.2.1.133 void dsp_normalize_nip (double * *dpDest*, const double * *dpSrc*, long *lCount*)

Definition at line 882 of file cdspop.cc.

References clDSPOp::Normalize().

6.2.1.134 float dsp_productf (const float * *fpSrc*, long *lCount*)

clDSPOp::Product

Definition at line 888 of file cdspop.cc.

References clDSPOp::Product().

6.2.1.135 double dsp_product (const double * *dpSrc*, long *lCount*)

Definition at line 894 of file cdspop.cc.

References clDSPOp::Product().

6.2.1.136 void dsp_reversef (float * *fpVect*, long *lCount*)

clDSPOp::Reverse

Definition at line 900 of file cdspop.cc.

References clDSPOp::Reverse().

6.2.1.137 void dsp_reverse (double * *dpVect*, long *lCount*)

Definition at line 906 of file cdspop.cc.

References clDSPOp::Reverse().

6.2.1.138 void dsp_reversef_nip (float ∗ *fpDest*, const float ∗ *fpSrc*, long *lCount*)

Definition at line 912 of file cdspop.cc.

References clDSPOp::Reverse().

6.2.1.139 void dsp_reverse_nip (double ∗ *dpDest*, const double ∗ *dpSrc*, long *lCount*)

Definition at line 918 of file cdspop.cc.

References clDSPOp::Reverse().

6.2.1.140 void dsp_scalef (float ∗ *fpVect*, long *lCount*)

clDSPOp::Scale

Definition at line 924 of file cdspop.cc.

References clDSPOp::Scale().

6.2.1.141 void dsp_scale (double ∗ *dpVect*, long *lCount*)

Definition at line 930 of file cdspop.cc.

References clDSPOp::Scale().

6.2.1.142 void dsp_scalef_nip (float ∗ *fpDest*, const float ∗ *fpSrc*, long *lCount*)

Definition at line 936 of file cdspop.cc.

References clDSPOp::Scale().

6.2.1.143 void dsp_scale_nip (double ∗ *dpDest*, const double ∗ *dpSrc*, long *lCount*)

Definition at line 942 of file cdspop.cc.

References clDSPOp::Scale().

6.2.1.144 void dsp_scale01f (float ∗ *fpVect*, long *lCount*)

clDSPOp::Scale01

Definition at line 948 of file cdspop.cc.

References clDSPOp::Scale01().

6.2.1.145 void dsp_scale01 (double $*$ *dpVect*, long *lCount*)

Definition at line 954 of file cdspop.cc.

References clDSPOp::Scale01().

6.2.1.146 void dsp_scale01f_nip (float $*$ *fpDest*, const float $*$ *fpSrc*, long *lCount*)

Definition at line 960 of file cdspop.cc.

References clDSPOp::Scale01().

6.2.1.147 void dsp_scale01_nip (double $*$ *dpDest*, const double $*$ *dpSrc*, long *lCount*)

Definition at line 966 of file cdspop.cc.

References clDSPOp::Scale01().

6.2.1.148 void dsp_sortf (float $*$ *fpVect*, long *lCount*)

clDSPOp::Sort

Definition at line 972 of file cdspop.cc.

References clDSPOp::Sort().

6.2.1.149 void dsp_sort (double $*$ *dpVect*, long *lCount*)

Definition at line 978 of file cdspop.cc.

References clDSPOp::Sort().

6.2.1.150 void dsp_sortl (long $*$ *lpVect*, long *lCount*)

Definition at line 984 of file cdspop.cc.

References clDSPOp::Sort().

6.2.1.151 void dsp_stddevf (float $*$ *fpStdDev*, float $*$ *fpMean*, const float $*$ *fpSrc*, long *lCount*)

clDSPOp::StdDev

Definition at line 990 of file cdspop.cc.

References clDSPOp::StdDev().

6.2.1.152 void dsp_stddev (double * *dpStdDev*, double * *dpMean*, const double * *dpSrc*, long *lCount*)

Definition at line 997 of file cdspop.cc.

References clDSPOp::StdDev().

6.2.1.153 float dsp_sumf (const float * *fpSrc*, long *lCount*)

clDSPOp::Sum

Definition at line 1004 of file cdspop.cc.

References clDSPOp::Sum().

6.2.1.154 double dsp_sum (const double * *dpSrc*, long *lCount*)

Definition at line 1010 of file cdspop.cc.

References clDSPOp::Sum().

6.2.1.155 void dsp_squaref (float * *fpVect*, long *lCount*)

clDSPOp::Square

Definition at line 1016 of file cdspop.cc.

References clDSPOp::Square().

6.2.1.156 void dsp_square (double * *dpVect*, long *lCount*)

Definition at line 1022 of file cdspop.cc.

References clDSPOp::Square().

6.2.1.157 void dsp_squaref_nip (float * *fpDest*, const float * *fpSrc*, long *lCount*)

Definition at line 1028 of file cdspop.cc.

References clDSPOp::Square().

6.2.1.158 void dsp_square_nip (double * *dpDest*, const double * *dpSrc*, long *lCount*)

Definition at line 1034 of file cdspop.cc.

References clDSPOp::Square().

6.2.1.159 void dsp_convertu8f (float * *fpDest*, const unsigned char * *ucpSrc*, long *lCount*)

clDSPOp::Convert

Definition at line 1040 of file cdspop.cc.

References clDSPOp::Convert().

6.2.1.160 void dsp_convertu8 (double * *dpDest*, const unsigned char * *ucpSrc*, long *lCount*)

Definition at line 1046 of file cdspop.cc.

References clDSPOp::Convert().

6.2.1.161 void dsp_converts16f (float * *fpDest*, const signed short * *ipSrc*, long *lCount*, int *i12bit*)

Definition at line 1052 of file cdspop.cc.

References clDSPOp::Convert().

6.2.1.162 void dsp_converts16 (double * *dpDest*, const signed short * *ipSrc*, long *lCount*, int *i12bit*)

Definition at line 1059 of file cdspop.cc.

References clDSPOp::Convert().

6.2.1.163 void dsp_converts32f (float * *fpDest*, const signed int * *ipSrc*, long *lCount*, int *i24bit*)

Definition at line 1066 of file cdspop.cc.

References clDSPOp::Convert().

6.2.1.164 void dsp_converts32 (double * *dpDest*, const signed int * *ipSrc*, long *lCount*, int *i24bit*)

Definition at line 1073 of file cdspop.cc.

References clDSPOp::Convert().

6.2.1.165 void dsp_convertd64f (float * *fpDest*, const double * *dpSrc*, long *lCount*)

Definition at line 1080 of file cdspop.cc.

References clDSPOp::Convert().

6.2.1.166 void dsp_convertf32 (double ∗ *dpDest*, const float ∗ *fpSrc*, long *lCount*)

Definition at line 1086 of file cdspop.cc.

References clDSPOp::Convert().

6.2.1.167 void dsp_convertf32c (unsigned char ∗ *ucpDest*, const float ∗ *fpSrc*, long *lCount*)

Definition at line 1092 of file cdspop.cc.

References clDSPOp::Convert().

6.2.1.168 void dsp_convertd64c (unsigned char ∗ *ucpDest*, const double ∗ *dpSrc*, long *lCount*)

Definition at line 1098 of file cdspop.cc.

References clDSPOp::Convert().

6.2.1.169 void dsp_convertf32s (signed short ∗ *ipDest*, const float ∗ *fpSrc*, long *lCount*, int *i12bit*)

Definition at line 1104 of file cdspop.cc.

References clDSPOp::Convert().

6.2.1.170 void dsp_convertd64s (signed short ∗ *ipDest*, const double ∗ *dpSrc*, long *lCount*, int *i12bit*)

Definition at line 1111 of file cdspop.cc.

References clDSPOp::Convert().

6.2.1.171 void dsp_convertf32i (signed int ∗ *ipDest*, const float ∗ *fpSrc*, long *lCount*, int *i24bit*)

Definition at line 1118 of file cdspop.cc.

References clDSPOp::Convert().

6.2.1.172 void dsp_convertd64i (signed int ∗ *ipDest*, const double ∗ *dpSrc*, long *lCount*, int *i24bit*)

Definition at line 1125 of file cdspop.cc.

References clDSPOp::Convert().

6.2.1.173 void dsp_cart2polarf (float ∗ *fpMagn*, float ∗ *fpPhase*, const float ∗ *fpReal*, const float ∗ *fpImag*, long *lCount*)

clDSPOp::CartToPolar

Definition at line 1132 of file cdspop.cc.

References clDSPOp::CartToPolar().

6.2.1.174 void dsp_cart2polar (double ∗ *dpMagn*, double ∗ *dpPhase*, const double ∗ *dpReal*, const double ∗ *dpImag*, long *lCount*)

Definition at line 1139 of file cdspop.cc.

References clDSPOp::CartToPolar().

6.2.1.175 void dsp_cart2polar2f (float ∗ *fpMagn*, float ∗ *fpPhase*, const stpSCplx *spCart*, long *lCount*)

Definition at line 1146 of file cdspop.cc.

References clDSPOp::CartToPolar(), and stpSCplx.

6.2.1.176 void dsp_cart2polar2 (double ∗ *dpMagn*, double ∗ *dpPhase*, const stpDCplx *spCart*, long *lCount*)

Definition at line 1153 of file cdspop.cc.

References clDSPOp::CartToPolar(), and stpDCplx.

6.2.1.177 void dsp_cart2polar3f (stpSPolar *spPolar*, const stpSCplx *spCart*, long *lCount*)

Definition at line 1160 of file cdspop.cc.

References clDSPOp::CartToPolar(), stpSCplx, and stpSPolar.

6.2.1.178 void dsp_cart2polar3 (stpDPolar *spPolar*, const stpDCplx *spCart*, long *lCount*)

Definition at line 1166 of file cdspop.cc.

References clDSPOp::CartToPolar(), stpDCplx, and stpDPolar.

6.2.1.179 void dsp_cart2polar4f (utpSCoord *upCoord*, long *lCount*)

Definition at line 1172 of file cdspop.cc.

References clDSPOp::CartToPolar(), and utpSCoord.

6.2.1.180 void dsp_cart2polar4 (utpDCoord *upCoord*, long *lCount*)

Definition at line 1178 of file cdspop.cc.

References clDSPOp::CartToPolar(), and utpDCoord.

6.2.1.181 void dsp_polar2cartf (float ∗ *fpReal*, float ∗ *fpImag*, const float ∗ *fpMagn*, const float ∗ *fpPhase*, long *lCount*)

clDSPOp::PolarToCart

Definition at line 1184 of file cdspop.cc.

References clDSPOp::PolarToCart().

6.2.1.182 void dsp_polar2cart (double ∗ *dpReal*, double ∗ *dpImag*, const double ∗ *dpMagn*, const double ∗ *dpPhase*, long *lCount*)

Definition at line 1191 of file cdspop.cc.

References clDSPOp::PolarToCart().

6.2.1.183 void dsp_polar2cart2f (stpSCplx *spCart*, const float ∗ *fpMagn*, const float ∗ *fpPhase*, long *lCount*)

Definition at line 1198 of file cdspop.cc.

References clDSPOp::PolarToCart(), and stpSCplx.

6.2.1.184 void dsp_polar2cart2 (stpDCplx *spCart*, const double ∗ *dpMagn*, const double ∗ *dpPhase*, long *lCount*)

Definition at line 1205 of file cdspop.cc.

References clDSPOp::PolarToCart(), and stpDCplx.

6.2.1.185 void dsp_polar2cart3f (stpSCplx *spCart*, const stpSPolar *spPolar*, long *lCount*)

Definition at line 1212 of file cdspop.cc.

References clDSPOp::PolarToCart(), stpSCplx, and stpSPolar.

6.2.1.186 void dsp_polar2cart3 (stpDCplx *spCart*, const stpDPolar *spPolar*, long *lCount*)

Definition at line 1218 of file cdspop.cc.

References clDSPOp::PolarToCart(), stpDCplx, and stpDPolar.

6.2.1.187 void dsp_polar2cart4f (utpSCoord *upCoord*, long *lCount*)

Definition at line 1224 of file cdspop.cc.

References clDSPOp::PolarToCart(), and utpSCoord.

6.2.1.188 void dsp_polar2cart4 (utpDCoord *upCoord*, long *lCount*)

Definition at line 1230 of file cdspop.cc.

References clDSPOp::PolarToCart(), and utpDCoord.

6.2.1.189 float dsp_crosscorrf (const float * *fpSrc1*, const float * *fpSrc2*, long *lCount*)

clDSPOp::CrossCorr

Definition at line 1236 of file cdspop.cc.

References clDSPOp::CrossCorr().

6.2.1.190 double dsp_crosscorr (const double * *dpSrc1*, const double * *dpSrc2*, long *lCount*)

Definition at line 1242 of file cdspop.cc.

References clDSPOp::CrossCorr().

6.2.1.191 float dsp_crosscorr2f (const float * *fpSrc1*, const float * *fpSrc2*, long *lDelay*, long *lCount*)

clDSPOp::DelCrossCorr

Definition at line 1248 of file cdspop.cc.

References clDSPOp::DelCrossCorr().

6.2.1.192 double dsp_crosscorr2 (const double * *dpSrc1*, const double * *dpSrc2*, long *lDelay*, long *lCount*)

Definition at line 1255 of file cdspop.cc.

References clDSPOp::DelCrossCorr().

6.2.1.193 void dsp_crosscorr3f (float * *fpDest*, const float * *fpSrc1*, const float * *fpSrc2*, long *lCount*, const long * *lpDelays*, long *lDelayCount*)

Definition at line 1262 of file cdspop.cc.

References clDSPOp::DelCrossCorr().

6.2.1.194 void dsp_crosscorr3 (double ∗ *dpDest*, const double ∗ *dpSrc1*, const double ∗ *dpSrc2*, long *lCount*, const long ∗ *lpDelays*, long *lDelayCount*)

Definition at line 1270 of file cdspop.cc.

References clDSPOp::DelCrossCorr().

6.2.1.195 float dsp_energyf (const float ∗ *fpSrc*, long *lCount*)

clDSPOp::Energy

Definition at line 1278 of file cdspop.cc.

References clDSPOp::Energy().

6.2.1.196 double dsp_energy (const double ∗ *dpSrc*, long *lCount*)

Definition at line 1284 of file cdspop.cc.

References clDSPOp::Energy().

6.2.1.197 void dsp_magnitudef (float ∗ *fpDest*, const stpSCplx *spSrc*, long *lCount*)

clDSPOp::Magnitude

Definition at line 1290 of file cdspop.cc.

References clDSPOp::Magnitude(), and stpSCplx.

6.2.1.198 void dsp_magnitude (double ∗ *dpDest*, const stpDCplx *spSrc*, long *lCount*)

Definition at line 1296 of file cdspop.cc.

References clDSPOp::Magnitude(), and stpDCplx.

6.2.1.199 void dsp_powerf (float ∗ *fpDest*, const stpSCplx *spSrc*, long *lCount*)

clDSPOp::Power

Definition at line 1302 of file cdspop.cc.

References clDSPOp::Power(), and stpSCplx.

6.2.1.200 void dsp_power (double ∗ *dpDest*, const stpDCplx *spSrc*, long *lCount*)

Definition at line 1308 of file cdspop.cc.

References clDSPOp::Power(), and stpDCplx.

6.2.1.201 void dsp_phasef (float * *fpDest*, const stpSCplx *spSrc*, long *lCount*)

clDSPOp::Phase

Definition at line 1314 of file cdspop.cc.

References clDSPOp::Phase(), and stpSCplx.

6.2.1.202 void dsp_phase (double * *dpDest*, const stpDCplx *spSrc*, long *lCount*)

Definition at line 1320 of file cdspop.cc.

References clDSPOp::Phase(), and stpDCplx.

6.2.1.203 void dsp_powerphasef (float * *fpPower*, float * *fpPhase*, const stpSCplx *spSrc*, long *lCount*)

clDSPOp::PowerPhase

Definition at line 1326 of file cdspop.cc.

References clDSPOp::PowerPhase(), and stpSCplx.

6.2.1.204 void dsp_powerphase (double * *dpPower*, double * *dpPhase*, const stpDCplx *spSrc*, long *lCount*)

Definition at line 1333 of file cdspop.cc.

References clDSPOp::PowerPhase(), and stpDCplx.

6.2.1.205 void dsp_decimatef (float * *fpDest*, const float * *fpSrc*, long *lFactor*, long *lCount*)

clDSPOp::Decimate

Definition at line 1340 of file cdspop.cc.

References clDSPOp::Decimate().

6.2.1.206 void dsp_decimate (double * *dpDest*, const double * *dpSrc*, long *lFactor*, long *lCount*)

Definition at line 1347 of file cdspop.cc.

References clDSPOp::Decimate().

6.2.1.207 void dsp_decimateavgf (float * *fpDest*, const float * *fpSrc*, long *lFactor*, long *lCount*)

clDSPOp::DecimateAvg

Definition at line 1354 of file cdspop.cc.

References clDSPOp::DecimateAvg().

6.2.1.208 void dsp_decimateavg (double * *dpDest*, const double * *dpSrc*, long *lFactor*, long *lCount*)

Definition at line 1361 of file cdspop.cc.

References clDSPOp::DecimateAvg().

6.2.1.209 void dsp_interpolatef (float * *fpDest*, const float * *fpSrc*, long *lFactor*, long *lCount*)

clDSPOp::Interpolate

Definition at line 1368 of file cdspop.cc.

References clDSPOp::Interpolate().

6.2.1.210 void dsp_interpolate (double * *dpDest*, const double * *dpSrc*, long *lFactor*, long *lCount*)

Definition at line 1375 of file cdspop.cc.

References clDSPOp::Interpolate().

6.2.1.211 void dsp_resamplef (float * *fpDest*, long *lDestCount*, const float * *fpSrc*, long *lSrcCount*)

clDSPOp::Resample

Definition at line 1382 of file cdspop.cc.

References clDSPOp::Resample().

6.2.1.212 void dsp_resample (double * *dpDest*, long *lDestCount*, const double * *dpSrc*, long *lSrcCount*)

Definition at line 1389 of file cdspop.cc.

References clDSPOp::Resample().

6.2.1.213 void dsp_resampleavgf (float * *fpDest*, long *lDestCount*, const float * *fpSrc*, long *lSrcCount*)

clDSPOp::ResampleAvg

Definition at line 1396 of file cdspop.cc.

References clDSPOp::ResampleAvg().

6.2.1.214 void dsp_resampleavg (double * *dpDest*, long *lDestCount*, const double * *dpSrc*, long *lSrcCount*)

Definition at line 1403 of file cdspop.cc.

References clDSPOp::ResampleAvg().

6.2.1.215 void dsp_interpolateavgf (float * *fpDest*, const float * *fpSrc*, long *lFactor*, long *lCount*)

clDSPOp::InterpolateAvg

Definition at line 1410 of file cdspop.cc.

References clDSPOp::InterpolateAvg().

6.2.1.216 void dsp_interpolateavg (double * *dpDest*, const double * *dpSrc*, long *lFactor*, long *lCount*)

Definition at line 1417 of file cdspop.cc.

References clDSPOp::InterpolateAvg().

6.2.1.217 float dsp_rmsf (const float * *fpSrc*, long *lCount*)

clDSPOp::RMS

Definition at line 1424 of file cdspop.cc.

References clDSPOp::RMS().

6.2.1.218 double dsp_rms (const double * *dpSrc*, long *lCount*)

Definition at line 1430 of file cdspop.cc.

References clDSPOp::RMS().

6.2.1.219 float dsp_variancef (float * *fpVariance*, float * *fpMean*, const float * *fpSrc*, long *lCount*)

clDSPOp::Variance

Definition at line 1436 of file cdspop.cc.

References clDSPOp::Variance().

6.2.1.220 double dsp_variance (double * *dpVariance*, double * *dpMean*, const double * *dpSrc*, long *lCount*)

Definition at line 1443 of file cdspop.cc.

References clDSPOp::Variance().

6.2.1.221 float dsp_peaklevelf (const float ∗ *fpSrc*, long *lCount*)

clDSPOp::PeakLevel

Definition at line 1450 of file cdspop.cc.

References clDSPOp::PeakLevel().

6.2.1.222 double dsp_peaklevel (const double ∗ *dpSrc*, long *lCount*)

Definition at line 1456 of file cdspop.cc.

References clDSPOp::PeakLevel().

6.2.1.223 void dsp_mixf (float ∗ *fpDest*, const float ∗ *fpSrc*, long *lCount*)

clDSPOp::Mix

Definition at line 1462 of file cdspop.cc.

References clDSPOp::Mix().

6.2.1.224 void dsp_mix (double ∗ *dpDest*, const double ∗ *dpSrc*, long *lCount*)

Definition at line 1468 of file cdspop.cc.

References clDSPOp::Mix().

6.2.1.225 void dsp_mix2f (float ∗ *fpDest*, const float ∗ *fpSrc1*, const float ∗ *fpSrc2*, long *lCount*)

Definition at line 1474 of file cdspop.cc.

References clDSPOp::Mix().

6.2.1.226 void dsp_mix2 (double ∗ *dpDest*, const double ∗ *dpSrc1*, const double ∗ *dpSrc2*, long *lCount*)

Definition at line 1481 of file cdspop.cc.

References clDSPOp::Mix().

6.2.1.227 void dsp_mixnf (float ∗ *fpDest*, const float ∗ *fpSrc*, long *lChCount*, long *lCount*)

Definition at line 1488 of file cdspop.cc.

References clDSPOp::Mix().

6.2.1.228 void dsp_mixn (double * *dpDest*, const double * *dpSrc*, long *lChCount*, long *lCount*)

Definition at line 1495 of file cdspop.cc.

References clDSPOp::Mix().

6.2.1.229 void dsp_extractf (float * *fpDest*, const float * *fpSrc*, long *lChannel*, long *lChCount*, long *lCount*)

clDSPOp::Extract

Definition at line 1502 of file cdspop.cc.

References clDSPOp::Extract().

6.2.1.230 void dsp_extract (double * *dpDest*, const double * *dpSrc*, long *lChannel*, long *lChCount*, long *lCount*)

Definition at line 1509 of file cdspop.cc.

References clDSPOp::Extract().

6.2.1.231 void dsp_packf (float * *fpDest*, const float * *fpSrc*, long *lChannel*, long *lChCount*, long *lCount*)

clDSPOp::Pack

Definition at line 1516 of file cdspop.cc.

References clDSPOp::Pack().

6.2.1.232 void dsp_pack (double * *dpDest*, const double * *dpSrc*, long *lChannel*, long *lChCount*, long *lCount*)

Definition at line 1523 of file cdspop.cc.

References clDSPOp::Pack().

6.2.1.233 void dsp_fftw_convertf2cf (stpSCplx *spDest*, const float * *fpSrc*, long *lCount*)

clDSPOp::FFTWConvert

Definition at line 1530 of file cdspop.cc.

References clDSPOp::FFTWConvert(), and stpSCplx.

6.2.1.234 void dsp_fftw_convertf2cd (stpDCplx *spDest*, const float * *fpSrc*, long *lCount*)

Definition at line 1536 of file cdspop.cc.

References clDSPOp::FFTWConvert(), and stpDCplx.

6.2.1.235 void dsp_fftw_convertd2cf (stpSCplx *spDest*, const double * *dpSrc*, long *lCount*)

Definition at line 1542 of file cdspop.cc.

References clDSPOp::FFTWConvert(), and stpSCplx.

6.2.1.236 void dsp_fftw_convertd2cd (stpDCplx *spDest*, const double * *dpSrc*, long *lCount*)

Definition at line 1548 of file cdspop.cc.

References clDSPOp::FFTWConvert(), and stpDCplx.

6.2.1.237 void dsp_fftw_convertcf2f (float * *fpDest*, const stpSCplx *spSrc*, long *lCount*)

Definition at line 1554 of file cdspop.cc.

References clDSPOp::FFTWConvert(), and stpSCplx.

6.2.1.238 void dsp_fftw_convertcd2f (float * *fpDest*, const stpDCplx *spSrc*, long *lCount*)

Definition at line 1560 of file cdspop.cc.

References clDSPOp::FFTWConvert(), and stpDCplx.

6.2.1.239 void dsp_fftw_convertcf2d (double * *dpDest*, const stpSCplx *spSrc*, long *lCount*)

Definition at line 1566 of file cdspop.cc.

References clDSPOp::FFTWConvert(), and stpSCplx.

6.2.1.240 void dsp_fftw_convertcd2d (double * *dpDest*, const stpDCplx *spSrc*, long *lCount*)

Definition at line 1572 of file cdspop.cc.

References clDSPOp::FFTWConvert(), and stpDCplx.

6.2.1.241 dsp_t dsp_new (void)

Creates new instance of DSP object.

Returns:

DSP object instance

Definition at line 1578 of file cdspop.cc.

References dsp_t.

6.2.1.242 void dsp_delete (dsp_t)

Deletes DSP object instance.

Parameters:

dspInst DSP object instance

Definition at line 1584 of file cdspop.cc.

References dsp_t.

6.2.1.243 void dsp_win_bartlettf (dsp_t *dspInst*, float ∗ *fpDest*, long *lCount*)

clDSPOp::WinBartlett

Definition at line 1590 of file cdspop.cc.

References dsp_t.

6.2.1.244 void dsp_win_bartlett (dsp_t *dspInst*, double ∗ *dpDest*, long *lCount*)

Definition at line 1596 of file cdspop.cc.

References dsp_t.

6.2.1.245 void dsp_win_blackmanf (dsp_t *dspInst*, float ∗ *fpDest*, long *lCount*, float *fAlpha*)

clDSPOp::WinBlackman

Definition at line 1602 of file cdspop.cc.

References dsp_t.

6.2.1.246 void dsp_win_blackman (dsp_t *dspInst*, double ∗ *dpDest*, long *lCount*, double *dAlpha*)

Definition at line 1609 of file cdspop.cc.

References dsp_t.

6.2.1.247 void dsp_win_blackman_harrisf (dsp_t *dspInst*, float ∗ *fpDest*, long *lCount*)

clDSPOp::WinBlackmanHarris

Definition at line 1616 of file cdspop.cc.

References dsp_t.

6.2.1.248 void dsp_win_blackman_harris (dsp_t *dspInst*, double ∗ *dpDest*, long *lCount*)

Definition at line 1622 of file cdspop.cc.

References dsp_t.

6.2.1.249 void dsp_win_cos_taperedf (dsp_t *dspInst*, float ∗ *fpDest*, long *lCount*)

clDSPOp::WinCosTapered

Definition at line 1628 of file cdspop.cc.

References dsp_t.

6.2.1.250 void dsp_win_cos_tapered (dsp_t *dspInst*, double ∗ *dpDest*, long *lCount*)

Definition at line 1634 of file cdspop.cc.

References dsp_t.

6.2.1.251 void dsp_win_exact_blackmanf (dsp_t *dspInst*, float ∗ *fpDest*, long *lCount*)

clDSPOp::WinExactBlackman

Definition at line 1640 of file cdspop.cc.

References dsp_t.

6.2.1.252 void dsp_win_exact_blackman (dsp_t *dspInst*, double ∗ *dpDest*, long *lCount*)

Definition at line 1646 of file cdspop.cc.

References dsp_t.

6.2.1.253 void dsp_win_flat_topf (dsp_t *dspInst*, float ∗ *fpDest*, long *lCount*)

clDSPOp::WinFlatTop

Definition at line 1652 of file cdspop.cc.

References dsp_t.

6.2.1.254 void dsp_win_flat_top (dsp_t *dspInst*, double ∗ *dpDest*, long *lCount*)

Definition at line 1658 of file cdspop.cc.

References dsp_t.

6.2.1.255 void dsp_win_generic_cosf (dsp_t *dspInst*, float ∗ *fpDest*, long *lCount*, const float ∗ *fpCoeff*, long *lCoeffCount*)

clDSPOp::WinGenericCos

Definition at line 1664 of file cdspop.cc.

References dsp_t.

6.2.1.256 void dsp_win_generic_cos (dsp_t *dspInst*, double ∗ *dpDest*, long *lCount*, const double ∗ *dpCoeff*, long *lCoeffCount*)

Definition at line 1672 of file cdspop.cc.

References dsp_t.

6.2.1.257 void dsp_win_hammingf (dsp_t *dspInst*, float ∗ *fpDest*, long *lCount*)

clDSPOp::WinHamming

Definition at line 1680 of file cdspop.cc.

References dsp_t.

6.2.1.258 void dsp_win_hamming (dsp_t *dspInst*, double ∗ *dpDest*, long *lCount*)

Definition at line 1686 of file cdspop.cc.

References dsp_t.

6.2.1.259 void dsp_win_hanningf (dsp_t *dspInst*, float ∗ *fpDest*, long *lCount*)

clDSPOp::WinHanning

Definition at line 1692 of file cdspop.cc.

References dsp_t.

6.2.1.260 void dsp_win_hanning (dsp_t *dspInst*, double ∗ *dpDest*, long *lCount*)

Definition at line 1698 of file cdspop.cc.

References dsp_t.

6.2.1.261 void dsp_win_kaiserf (dsp_t *dspInst*, float ∗ *fpDest*, float *fBeta*, long *lCount*)

clDSPOp::WinKaiser

Definition at line 1704 of file cdspop.cc.

References dsp_t.

6.2.1.262 void dsp_win_kaiser (dsp_t *dspInst*, double ∗ *dpDest*, double *dBeta*, long *lCount*)

Definition at line 1710 of file cdspop.cc.

References dsp_t.

6.2.1.263 void dsp_win_kaiser_besself (dsp_t *dspInst*, float ∗ *fpDest*, float *fAlpha*, long *lCount*)

clDSPOp::WinKaiserBessel

Definition at line 1716 of file cdspop.cc.

References dsp_t.

6.2.1.264 void dsp_win_kaiser_bessel (dsp_t *dspInst*, double ∗ *dpDest*, double *dAlpha*, long *lCount*)

Definition at line 1723 of file cdspop.cc.

References dsp_t.

6.2.1.265 void dsp_win_tukeyf (dsp_t *dspInst*, float ∗ *fpDest*, long *lCount*)

clDSPOp::WinTukey

Definition at line 1730 of file cdspop.cc.

References dsp_t.

6.2.1.266 void dsp_win_tukey (dsp_t *dspInst*, double ∗ *dpDest*, long *lCount*)

Definition at line 1736 of file cdspop.cc.

References dsp_t.

6.2.1.267 void dsp_win_dolph_chebyshevf (dsp_t *dspInst*, float ∗ *fpDest*, float *fGamma*, long *lCount*)

clDSPOp::WinDolphChebyshev

Definition at line 1742 of file cdspop.cc.

References dsp_t.

6.2.1.268 void dsp_win_dolph_chebyshev (dsp_t *dspInst*, double ∗ *dpDest*, double *dGamma*, long *lCount*)

Definition at line 1749 of file cdspop.cc.

References dsp_t.

6.2.1.269 long dsp_rebufferf (dsp_t *dspInst*, float ∗ *fpDest*, const float ∗ *fpSrc*, long *lDestCount*, long *lSrcCount*)

clDSPOp::ReBuffer

Definition at line 1756 of file cdspop.cc.

References dsp_t.

6.2.1.270 long dsp_rebuffer (dsp_t *dspInst*, double ∗ *dpDest*, const double ∗ *dpSrc*, long *lDestCount*, long *lSrcCount*)

Definition at line 1764 of file cdspop.cc.

References dsp_t.

6.2.1.271 float dsp_deg2radf (dsp_t *dspInst*, float *fSource*)

clDSPOp::DegToRad

Definition at line 1772 of file cdspop.cc.

References dsp_t.

6.2.1.272 double dsp_deg2rad (dsp_t *dspInst*, double *dSource*)

Definition at line 1778 of file cdspop.cc.

References dsp_t.

6.2.1.273 float dsp_rad2degf (dsp_t *dspInst*, float *fSource*)

clDSPOp::RadToDeg

Definition at line 1784 of file cdspop.cc.

References dsp_t.

6.2.1.274 double dsp_rad2deg (dsp_t *dspInst*, double *dSource*)

Definition at line 1790 of file cdspop.cc.

References dsp_t.

6.2.1.275 void dsp_fir_allocatef (dsp_t *dspInst*, const float * *fpCoeff*, long *lCount*)

clDSPOp::FIRAllocate

Definition at line 1796 of file cdspop.cc.

References dsp_t.

6.2.1.276 void dsp_fir_allocate (dsp_t *dspInst*, const double * *dpCoeff*, long *lCount*)

Definition at line 1802 of file cdspop.cc.

References dsp_t.

6.2.1.277 void dsp_fir_free (dsp_t *dspInst*)

clDSPOp::FIRFree

Definition at line 1808 of file cdspop.cc.

References dsp_t.

6.2.1.278 void dsp_fir_filterf (dsp_t *dspInst*, float * *fpVect*, long *lCount*)

clDSPOp::FIRFilter

Definition at line 1814 of file cdspop.cc.

References dsp_t.

6.2.1.279 void dsp_fir_filter (dsp_t *dspInst*, double * *dpVect*, long *lCount*)

Definition at line 1820 of file cdspop.cc.

References dsp_t.

6.2.1.280 void dsp_fir_filterf_nip (dsp_t *dspInst*, float * *fpDest*, const float * *fpSrc*, long *lCount*)

Definition at line 1826 of file cdspop.cc.

References dsp_t.

6.2.1.281 void dsp_fir_filter_nip (dsp_t *dspInst*, double ∗ *dpDest*, const double ∗ *dpSrc*, long *lCount*)

Definition at line 1833 of file cdspop.cc.

References dsp_t.

6.2.1.282 void dsp_fir_filterf_fst (dsp_t *dspInst*, float ∗ *fpDest*, float ∗ *fpSrc*, long *lCount*)

clDSPOp::FIRFilterF

Definition at line 1840 of file cdspop.cc.

References dsp_t.

6.2.1.283 void dsp_fir_filter_fst (dsp_t *dspInst*, double ∗ *dpDest*, double ∗ *dpSrc*, long *lCount*)

Definition at line 1847 of file cdspop.cc.

References dsp_t.

6.2.1.284 void dsp_iir_initf (dsp_t *dspInst*, const float ∗ *fpCoeffs*)

clDSPOp::IIRInitialize

Definition at line 1854 of file cdspop.cc.

References dsp_t.

6.2.1.285 void dsp_iir_init (dsp_t *dspInst*, const double ∗ *dpCoeffs*)

Definition at line 1860 of file cdspop.cc.

References dsp_t.

6.2.1.286 void dsp_iir_filterf (dsp_t *dspInst*, float ∗ *fpVect*, long *lCount*)

clDSPOp::IIRFilter

Definition at line 1866 of file cdspop.cc.

References dsp_t.

6.2.1.287 void dsp_iir_filter (dsp_t *dspInst*, double ∗ *dpVect*, long *lCount*)

Definition at line 1872 of file cdspop.cc.

References dsp_t.

6.2.1.288 void dsp_iir_filterf_nip (dsp_t *dspInst*, float * *fpDest*, const float * *fpSrc*, long *lCount*)

Definition at line 1878 of file cdspop.cc.

References dsp_t.

6.2.1.289 void dsp_iir_filter_nip (dsp_t *dspInst*, double * *dpDest*, const double * *dpSrc*, long *lCount*)

Definition at line 1885 of file cdspop.cc.

References dsp_t.

6.2.1.290 void dsp_iir_clear (dsp_t *dspInst*)

clDSPOp::IIRClear

Definition at line 1892 of file cdspop.cc.

References dsp_t.

6.2.1.291 void dsp_fft_init (dsp_t *dspInst*, long *lSize*, int *iReal*)

clDSPOp::FFTInitialize

Definition at line 1898 of file cdspop.cc.

References dsp_t.

6.2.1.292 void dsp_fft_uninit (dsp_t *dspInst*)

clDSPOp::FFTUninitialize

Definition at line 1904 of file cdspop.cc.

References dsp_t.

6.2.1.293 void dsp_fftf (dsp_t *dspInst*, stpSCplx *spDest*, float * *fpSrc*)

clDSPOp::FFTi

Definition at line 1910 of file cdspop.cc.

References dsp_t, and stpSCplx.

6.2.1.294 void dsp_fft (dsp_t *dspInst*, stpDCplx *spDest*, double $*$ *dpSrc*)

Definition at line 1916 of file cdspop.cc.

References dsp_t, and stpDCplx.

6.2.1.295 void dsp_fftf_nip (dsp_t *dspInst*, stpSCplx *spDest*, const float $*$ *fpSrc*)

clDSPOp::FFTo

Definition at line 1922 of file cdspop.cc.

References dsp_t, and stpSCplx.

6.2.1.296 void dsp_fft_nip (dsp_t *dspInst*, stpDCplx *spDest*, const double $*$ *dpSrc*)

Definition at line 1928 of file cdspop.cc.

References dsp_t, and stpDCplx.

6.2.1.297 void dsp_cfftf_nip (dsp_t *dspInst*, stpSCplx *spDest*, const stpSCplx *spSrc*)

Definition at line 1934 of file cdspop.cc.

References dsp_t, and stpSCplx.

6.2.1.298 void dsp_cfft_nip (dsp_t *dspInst*, stpDCplx *spDest*, const stpDCplx *spSrc*)

Definition at line 1940 of file cdspop.cc.

References dsp_t, and stpDCplx.

6.2.1.299 void dsp_ifftf_nip (dsp_t *dspInst*, float $*$ *fpDest*, const stpSCplx *spSrc*)

clDSPOp::IFFTo

Definition at line 1946 of file cdspop.cc.

References dsp_t, and stpSCplx.

6.2.1.300 void dsp_ifft_nip (dsp_t *dspInst*, double $*$ *dpDest*, const stpDCplx *spSrc*)

Definition at line 1952 of file cdspop.cc.

References dsp_t, and stpDCplx.

6.2.1.301 void dsp_cifftf_nip (dsp_t *dspInst*, stpSCplx *spDest*, const stpSCplx *spSrc*)

Definition at line 1958 of file cdspop.cc.

References dsp_t, and stpSCplx.

6.2.1.302 void dsp_cifft_nip (dsp_t *dspInst*, stpDCplx *spDest*, const stpDCplx *spSrc*)

Definition at line 1964 of file cdspop.cc.

References dsp_t, and stpDCplx.

6.2.1.303 void dsp_dct_init (dsp_t *dspInst*, long *lSize*)

clDSPOp::DCTInitialize

Definition at line 1970 of file cdspop.cc.

References dsp_t.

6.2.1.304 void dsp_dct_uninit (dsp_t *dspInst*)

clDSPOp::DCTUninitialize

Definition at line 1976 of file cdspop.cc.

References dsp_t.

6.2.1.305 void dsp_dctf (dsp_t *dspInst*, float ∗ *fpVect*)

clDSPOp::DCTi

Definition at line 1982 of file cdspop.cc.

References dsp_t.

6.2.1.306 void dsp_dct (dsp_t *dspInst*, double ∗ *dpVect*)

Definition at line 1988 of file cdspop.cc.

References dsp_t.

6.2.1.307 void dsp_dctf_nip (dsp_t *dspInst*, float ∗ *fpDest*, const float ∗ *fpSrc*)

clDSPOp::DCTo

Definition at line 1994 of file cdspop.cc.

References dsp_t.

6.2.1.308 void dsp_dct_nip (dsp_t *dspInst*, double * *dpDest*, const double * *dpSrc*)

Definition at line 2000 of file cdspop.cc.

References dsp_t.

6.2.1.309 void dsp_idctf (dsp_t *dspInst*, float * *fpVect*)

clDSPOp::IDCTi

Definition at line 2006 of file cdspop.cc.

References dsp_t.

6.2.1.310 void dsp_idct (dsp_t *dspInst*, double * *dpVect*)

Definition at line 2012 of file cdspop.cc.

References dsp_t.

6.2.1.311 void dsp_idctf_nip (dsp_t *dspInst*, float * *fpDest*, const float * *fpSrc*)

clDSPOp::IDCTo

Definition at line 2018 of file cdspop.cc.

References dsp_t.

6.2.1.312 void dsp_idct_nip (dsp_t *dspInst*, double * *dpDest*, const double * *dpSrc*)

Definition at line 2024 of file cdspop.cc.

References dsp_t.

6.2.1.313 dsp_iircas_t dsp_iir_cas_new (void)

Creates new instance of cascaded IIR object.

Returns:
Cascade IIR instance

Definition at line 2033 of file cdspop.cc.

References dsp_iircas_t.

6.2.1.314 void dsp_iir_cas_delete (dsp_iircas_t)

Deletes cascaded IIR object instance.

Parameters:

dspiircasInst Cascaded IIR instance

Definition at line 2039 of file cdspop.cc.

References dsp_iircas_t.

6.2.1.315 int dsp_iir_cas_initf (dsp_iircas_t *dspiircasInst*, const float *fpCoeffs*[ ][5], long *lStageCount*)

clIIRCascade::Initialize

Definition at line 2045 of file cdspop.cc.

References dsp_iircas_t.

6.2.1.316 int dsp_iir_cas_init (dsp_iircas_t *dspiircasInst*, const double *dpCoeffs*[ ][5], long *lStageCount*)

Definition at line 2054 of file cdspop.cc.

References dsp_iircas_t.

6.2.1.317 void dsp_iir_cas_uninit (dsp_iircas_t *dspiircasInst*)

Definition at line 2063 of file cdspop.cc.

References dsp_iircas_t.

6.2.1.318 void dsp_iir_cas_processf (dsp_iircas_t *dspiircasInst*, float ∗ *fpVect*, long *lCount*)

Definition at line 2069 of file cdspop.cc.

References dsp_iircas_t.

6.2.1.319 void dsp_iir_cas_process (dsp_iircas_t *dspiircasInst*, double ∗ *dpVect*, long *lCount*)

Definition at line 2076 of file cdspop.cc.

References dsp_iircas_t.

6.2.1.320 void dsp_iir_cas_processf_nip (dsp_iircas_t *dspiircasInst*, float ∗ *fpDest*, const float ∗ *fpSrc*, long *lCount*)

Definition at line 2083 of file cdspop.cc.

References dsp_iircas_t.

6.2.1.321 void dsp_iir_cas_process_nip (dsp_iircas_t *dspiircasInst*, double * *dpDest*, const double * *dpSrc*, long *lCount*)

Definition at line 2090 of file cdspop.cc.

References dsp_iircas_t.

6.2.1.322 void dsp_iir_cas_clear (dsp_iircas_t *dspiircasInst*)

Definition at line 2097 of file cdspop.cc.

References dsp_iircas_t.

6.2.1.323 dsp_decfft_t dsp_dec_fft_new (void)

Creates new instance of FFT decimator object.

Returns:

FFT decimator instance

Definition at line 2106 of file cdspop.cc.

References dsp_decfft_t.

6.2.1.324 void dsp_dec_fft_delete (dsp_decfft_t)

Deletes FFT decimator object instance.

Parameters:

dspdecfftInst FFT decimator instance

Definition at line 2112 of file cdspop.cc.

References dsp_decfft_t.

6.2.1.325 int dsp_dec_fft_initf (dsp_decfft_t *dspdecfftInst*, long *lDecFact*, long *lFiltSize*, int *iHighPass*)

clFFTDecimator::Initialize

Definition at line 2118 of file cdspop.cc.

References dsp_decfft_t.

6.2.1.326 int dsp_dec_fft_init (dsp_decfft_t *dspdecfftInst*, long *lDecFact*, long *lFiltSize*, int *iHighPass*)

Definition at line 2129 of file cdspop.cc.

References dsp_decfft_t.

6.2.1.327 void dsp_dec_fft_uninit (dsp_decfft_t *dspdecfftInst*)

clFFTDecimator::Uninitialize

Definition at line 2140 of file cdspop.cc.

References dsp_decfft_t.

6.2.1.328 void dsp_dec_fft_putf (dsp_decfft_t *dspdecfftInst*, const float ∗ *fpSrcData*, long *lSrcCount*)

clFFTDecimator::Put

Definition at line 2146 of file cdspop.cc.

References dsp_decfft_t.

6.2.1.329 void dsp_dec_fft_put (dsp_decfft_t *dspdecfftInst*, const double ∗ *dpSrcData*, long *lSrcCount*)

Definition at line 2153 of file cdspop.cc.

References dsp_decfft_t.

6.2.1.330 int dsp_dec_fft_getf (dsp_decfft_t *dspdecfftInst*, float ∗ *fpDestData*, long *lDestCount*)

clFFTDecimator::Get

Definition at line 2160 of file cdspop.cc.

References dsp_decfft_t.

6.2.1.331 int dsp_dec_fft_get (dsp_decfft_t *dspdecfftInst*, double ∗ *dpDestData*, long *lDestCount*)

Definition at line 2169 of file cdspop.cc.

References dsp_decfft_t.

6.2.1.332 dsp_decfir_t dsp_dec_fir_new (void)

Creates new instance of FIR decimator object.

Returns:
FIR decimator instance

Definition at line 2181 of file cdspop.cc.

References dsp_decfir_t.

6.2.1.333 void dsp_dec_fir_delete (dsp_decfir_t)

Deletes FIR decimator object instance.

Parameters:
dspdecfirInst FIR decimator instance

Definition at line 2187 of file cdspop.cc.

References dsp_decfir_t.

6.2.1.334 int dsp_dec_fir_initf (dsp_decfir_t *dspdecfirInst*, long *lDecFact*, int *iHighPass*)

clFIRDecimator::Initialize

Definition at line 2193 of file cdspop.cc.

References dsp_decfir_t.

6.2.1.335 int dsp_dec_fir_init (dsp_decfir_t *dspdecfirInst*, long *lDecFact*, int *iHighPass*)

Definition at line 2203 of file cdspop.cc.

References dsp_decfir_t.

6.2.1.336 void dsp_dec_fir_uninit (dsp_decfir_t *dspdecfirInst*)

clFIRDecimator::Uninitialize

Definition at line 2213 of file cdspop.cc.

References dsp_decfir_t.

6.2.1.337 void dsp_dec_fir_putf (dsp_decfir_t *dspdecfirInst*, const float ∗ *fpSrcData*, long *lSrcCount*)

clFIRDecimator::Put

Definition at line 2219 of file cdspop.cc.

References dsp_decfir_t.

6.2.1.338 void dsp_dec_fir_put (dsp_decfir_t *dspdecfirInst*, const double ∗ *dpSrcData*, long *lSrcCount*)

Definition at line 2226 of file cdspop.cc.

References dsp_decfir_t.

6.2.1.339 int dsp_dec_fir_getf (dsp_decfir_t *dspdecfirInst*, float ∗ *fpDestData*, long *lDestCount*)

clFIRDecimator::Get

Definition at line 2233 of file cdspop.cc.

References dsp_decfir_t.

6.2.1.340 int dsp_dec_fir_get (dsp_decfir_t *dspdecfirInst*, double ∗ *dpDestData*, long *lDestCount*)

Definition at line 2242 of file cdspop.cc.

References dsp_decfir_t.

6.2.1.341 dsp_deciir_t dsp_dec_iir_new (void)

Creates new instance of IIR decimator object.

Returns:
IIR decimator instance

Definition at line 2254 of file cdspop.cc.

References dsp_deciir_t.

6.2.1.342 void dsp_dec_iir_delete (dsp_deciir_t)

Deletes IIR decimator object instance.

Parameters:
dspdeciirInst IIR decimator instance

Definition at line 2260 of file cdspop.cc.

References dsp_deciir_t.

6.2.1.343 int dsp_dec_iir_initf (dsp_deciir_t *dspdeciirInst*, long *lDecFact*, int *iHighPass*)

clIIRDecimator::Initialize

Definition at line 2266 of file cdspop.cc.

References dsp_deciir_t.

6.2.1.344 int dsp_dec_iir_init (dsp_deciir_t *dspdeciirInst*, long *lDecFact*, int *iHighPass*)

Definition at line 2276 of file cdspop.cc.

References dsp_deciir_t.

6.2.1.345 void dsp_dec_iir_uninit (dsp_deciir_t *dspdeciirInst*)

clIIRDecimator::Uninitialize

Definition at line 2286 of file cdspop.cc.

References dsp_deciir_t.

6.2.1.346 void dsp_dec_iir_putf (dsp_deciir_t *dspdeciirInst*, const float * *fpSrcData*, long *lSrcCount*)

clIIRDecimator::Put

Definition at line 2292 of file cdspop.cc.

References dsp_deciir_t.

6.2.1.347 void dsp_dec_iir_put (dsp_deciir_t *dspdeciirInst*, const double * *dpSrcData*, long *lSrcCount*)

Definition at line 2299 of file cdspop.cc.

References dsp_deciir_t.

6.2.1.348 int dsp_dec_iir_getf (dsp_deciir_t *dspdeciirInst*, float * *fpDestData*, long *lDestCount*)

clIIRDecimator::Get

Definition at line 2306 of file cdspop.cc.

References dsp_deciir_t.

6.2.1.349 int dsp_dec_iir_get (dsp_deciir_t *dspdeciirInst*, double * *dpDestData*, long *lDestCount*)

Definition at line 2315 of file cdspop.cc.

References dsp_deciir_t.

6.2.1.350 dsp_decrec_t dsp_dec_rec_new (void)

Creates new instance of recursive decimator object.

Returns:
Recursive decimator instance

Definition at line 2327 of file cdspop.cc.

References dsp_decrec_t.

6.2.1.351 void dsp_dec_rec_delete (dsp_decrec_t)

Deletes recursive decimator object instance.

Parameters:
dspdecrecInst Recursive decimator instance

Definition at line 2333 of file cdspop.cc.

References dsp_decrec_t.

6.2.1.352 int dsp_dec_rec_initf (dsp_decrec_t *dspdecrecInst*, long *lDecFact*, long *lFiltSize*, float *fBandCenter*, int *iFilterType*)

clRecDecimator::Initialize

Definition at line 2339 of file cdspop.cc.

References dsp_decrec_t.

6.2.1.353 int dsp_dec_rec_init (dsp_decrec_t *dspdecrecInst*, long *lDecFact*, long *lFiltSize*, double *dBandCenter*, int *iFilterType*)

Definition at line 2351 of file cdspop.cc.

References dsp_decrec_t.

6.2.1.354 void dsp_dec_rec_uninit (dsp_decrec_t *dspdecrecInst*)

clRecDecimator::Uninitialize

Definition at line 2363 of file cdspop.cc.

References dsp_decrec_t.

6.2.1.355 void dsp_dec_rec_putf (dsp_decrec_t *dspdecrecInst*, const float * *fpSrcData*, long *lSrcCount*)

clRecDecimator::Put

Definition at line 2369 of file cdspop.cc.

References dsp_decrec_t.

6.2.1.356 void dsp_dec_rec_put (dsp_decrec_t *dspdecrecInst*, const double * *dpSrcData*, long *lSrcCount*)

Definition at line 2376 of file cdspop.cc.

References dsp_decrec_t.

6.2.1.357 int dsp_dec_rec_getf (dsp_decrec_t *dspdecrecInst*, float * *fpDestData*, long *lDestCount*)

clRecDecimator::Get

Definition at line 2383 of file cdspop.cc.

References dsp_decrec_t.

6.2.1.358 int dsp_dec_rec_get (dsp_decrec_t *dspdecrecInst*, double * *dpDestData*, long *lDestCount*)

Definition at line 2392 of file cdspop.cc.

References dsp_decrec_t.

6.2.1.359 dsp_intfft_t dsp_int_fft_new (void)

Creates new instance of FFT interpolator object.

Returns:
FFT interpolator instance

Definition at line 2404 of file cdspop.cc.

References dsp_intfft_t.

6.2.1.360 void dsp_int_fft_delete (dsp_intfft_t)

Deletes FFT interpolator object instance.

Parameters:
dspintfftInst FFT interpolator instance

Definition at line 2410 of file cdspop.cc.

References dsp_intfft_t.

6.2.1.361 int dsp_int_fft_initf (dsp_intfft_t *dspintfftInst*, long *lIntFact*, long *lFiltSize*, int *iHighPass*)

clFFTInterpolator::Initialize

Definition at line 2416 of file cdspop.cc.

References dsp_intfft_t.

6.2.1.362 int dsp_int_fft_init (dsp_intfft_t *dspintfftInst*, long *lIntFact*, long *lFiltSize*, int *iHighPass*)

Definition at line 2427 of file cdspop.cc.

References dsp_intfft_t.

6.2.1.363 void dsp_int_fft_uninit (dsp_intfft_t *dspintfftInst*)

clFFTInterpolator::Uninitialize

Definition at line 2438 of file cdspop.cc.

References dsp_intfft_t.

6.2.1.364 void dsp_int_fft_putf (dsp_intfft_t *dspintfftInst*, const float ∗ *fpSrcData*, long *lSrcCount*)

clFFTInterpolator::Put

Definition at line 2444 of file cdspop.cc.

References dsp_intfft_t.

6.2.1.365 void dsp_int_fft_put (dsp_intfft_t *dspintfftInst*, const double ∗ *dpSrcData*, long *lSrcCount*)

Definition at line 2451 of file cdspop.cc.

References dsp_intfft_t.

6.2.1.366 int dsp_int_fft_getf (dsp_intfft_t *dspintfftInst*, float ∗ *fpDestData*, long *lDestCount*)

clFFTInterpolator::Get

Definition at line 2458 of file cdspop.cc.

References dsp_intfft_t.

6.2.1.367 int dsp_int_fft_get (dsp_intfft_t *dspintfftInst*, double ∗ *dpDestData*, long *lDestCount*)

Definition at line 2467 of file cdspop.cc.

References dsp_intfft_t.

6.2.1.368 dsp_intfir_t dsp_int_fir_new (void)

Creates new instance of FIR interpolator object.

Returns:

FIR interpolator instance

Definition at line 2479 of file cdspop.cc.

References dsp_intfir_t.

6.2.1.369 void dsp_int_fir_delete (dsp_intfir_t)

Deletes FIR interpolator object instance.

Parameters:

dspintfirInst FIR interpolator instance

Definition at line 2485 of file cdspop.cc.

References dsp_intfir_t.

6.2.1.370 int dsp_int_fir_initf (dsp_intfir_t *dspintfirInst*, long *lIntFact*, int *iHighPass*)

clFIRInterpolator::Initialize

Definition at line 2491 of file cdspop.cc.

References dsp_intfir_t.

6.2.1.371 int dsp_int_fir_init (dsp_intfir_t *dspintfirInst*, long *lIntFact*, int *iHighPass*)

Definition at line 2501 of file cdspop.cc.

References dsp_intfir_t.

6.2.1.372 void dsp_int_fir_uninit (dsp_intfir_t *dspintfirInst*)

clFIRInterpolator::Uninitialize

Definition at line 2511 of file cdspop.cc.

References dsp_intfir_t.

6.2.1.373 void dsp_int_fir_putf (dsp_intfir_t *dspintfirInst*, const float * *fpSrcData*, long *lSrcCount*)

clFIRInterpolator::Put

Definition at line 2517 of file cdspop.cc.

References dsp_intfir_t.

6.2.1.374 void dsp_int_fir_put (dsp_intfir_t *dspintfirInst*, const double ∗ *dpSrcData*, long *lSrcCount*)

Definition at line 2524 of file cdspop.cc.

References dsp_intfir_t.

6.2.1.375 int dsp_int_fir_getf (dsp_intfir_t *dspintfirInst*, float ∗ *fpDestData*, long *lDestCount*)

clFIRInterpolator::Get

Definition at line 2531 of file cdspop.cc.

References dsp_intfir_t.

6.2.1.376 int dsp_int_fir_get (dsp_intfir_t *dspintfirInst*, double ∗ *dpDestData*, long *lDestCount*)

Definition at line 2540 of file cdspop.cc.

References dsp_intfir_t.

6.2.1.377 dsp_intiir_t dsp_int_iir_new (void)

Creates new instance of IIR interpolator object.

Returns:

IIR interpolator instance

Definition at line 2552 of file cdspop.cc.

References dsp_intiir_t.

6.2.1.378 void dsp_int_iir_delete (dsp_intiir_t)

Deletes IIR interpolator object instance.

Parameters:

dspintiirInst IIR interpolator instance

Definition at line 2558 of file cdspop.cc.

References dsp_intiir_t.

6.2.1.379 int dsp_int_iir_initf (dsp_intiir_t *dspintiirInst*, long *lIntFact*, int *iHighPass*)

clIIRInterpolator::Initialize

Definition at line 2564 of file cdspop.cc.

References dsp_intiir_t.

6.2.1.380 int dsp_int_iir_init (dsp_intiir_t *dspintiirInst*, long *lIntFact*, int *iHighPass*)

Definition at line 2574 of file cdspop.cc.

References dsp_intiir_t.

6.2.1.381 void dsp_int_iir_uninit (dsp_intiir_t *dspintiirInst*)

clIIRInterpolator::Uninitialize

Definition at line 2584 of file cdspop.cc.

References dsp_intiir_t.

6.2.1.382 void dsp_int_iir_putf (dsp_intiir_t *dspintiirInst*, const float ∗ *fpSrcData*, long *lSrcCount*)

clIIRInterpolator::Put

Definition at line 2590 of file cdspop.cc.

References dsp_intiir_t.

6.2.1.383 void dsp_int_iir_put (dsp_intiir_t *dspintiirInst*, const double ∗ *dpSrcData*, long *lSrcCount*)

Definition at line 2597 of file cdspop.cc.

References dsp_intiir_t.

6.2.1.384 int dsp_int_iir_getf (dsp_intiir_t *dspintiirInst*, float ∗ *fpDestData*, long *lDestCount*)

clIIRInterpolator::Get

Definition at line 2604 of file cdspop.cc.

References dsp_intiir_t.

6.2.1.385 int dsp_int_iir_get (dsp_intiir_t *dspintiirInst*, double * *dpDestData*, long *lDestCount*)

Definition at line 2613 of file cdspop.cc.

References dsp_intiir_t.

6.2.1.386 dsp_intrec_t dsp_int_rec_new (void)

Creates new instance of recursive interpolator object.

Returns:

Recursive interpolator instance

Definition at line 2625 of file cdspop.cc.

References dsp_intrec_t.

6.2.1.387 void dsp_int_rec_delete (dsp_intrec_t)

Deletes recursive interpolator object instance.

Parameters:

dspdecrecInst Recursive interpolator instance

Definition at line 2631 of file cdspop.cc.

References dsp_intrec_t.

6.2.1.388 int dsp_int_rec_initf (dsp_intrec_t *dspintrecInst*, long *lIntFact*, long *lFiltSize*, float *fBandCenter*, int *iFilterType*)

clRecInterpolator::Initialize

Definition at line 2637 of file cdspop.cc.

References dsp_intrec_t.

6.2.1.389 int dsp_int_rec_init (dsp_intrec_t *dspintrecInst*, long *lIntFact*, long *lFiltSize*, double *dBandCenter*, int *iFilterType*)

Definition at line 2649 of file cdspop.cc.

References dsp_intrec_t.

6.2.1.390 void dsp_int_rec_uninit (dsp_intrec_t *dspintrecInst*)

clRecInterpolator::Uninitialize

Definition at line 2661 of file cdspop.cc.

References dsp_intrec_t.

6.2.1.391 void dsp_int_rec_putf (dsp_intrec_t *dspintrecInst*, const float * *fpSrcData*, long *lSrcCount*)

clRecInterpolator::Put

Definition at line 2667 of file cdspop.cc.

References dsp_intrec_t.

6.2.1.392 void dsp_int_rec_put (dsp_intrec_t *dspintrecInst*, const double * *dpSrcData*, long *lSrcCount*)

Definition at line 2674 of file cdspop.cc.

References dsp_intrec_t.

6.2.1.393 int dsp_int_rec_getf (dsp_intrec_t *dspintrecInst*, float * *fpDestData*, long *lDestCount*)

clRecInterpolator::Get

Definition at line 2681 of file cdspop.cc.

References dsp_intrec_t.

6.2.1.394 int dsp_int_rec_get (dsp_intrec_t *dspintrecInst*, double * *dpDestData*, long *lDestCount*)

Definition at line 2690 of file cdspop.cc.

References dsp_intrec_t.

6.2.1.395 dsp_filter_t dsp_filter_new (void)

Creates new instance of FFT filter object.

Returns:
FFT filter instance

Definition at line 2702 of file cdspop.cc.

References dsp_filter_t.

6.2.1.396 void dsp_filter_delete (dsp_filter_t)

Deletes instance of FFT filter object.

Parameters:

dspfilterInst FFT filter instance

Definition at line 2708 of file cdspop.cc.

References dsp_filter_t.

6.2.1.397 int dsp_filter_initf (dsp_filter_t *dspfilterInst*, long *lWindowSize*)

clFilter::Initialize

Definition at line 2714 of file cdspop.cc.

References dsp_filter_t.

6.2.1.398 int dsp_filter_init (dsp_filter_t *dspfilterInst*, long *lWindowSize*)

Definition at line 2724 of file cdspop.cc.

References dsp_filter_t.

6.2.1.399 int dsp_filter_init2f (dsp_filter_t *dspfilterInst*, long *lWindowSize*, const float ∗ *fpFiltCoeffs*, float *fOverlap*, float *fBeta*)

Definition at line 2734 of file cdspop.cc.

References dsp_filter_t.

6.2.1.400 int dsp_filter_init2 (dsp_filter_t *dspfilterInst*, long *lWindowSize*, const double ∗ *dpFiltCoeffs*, double *dOverlap*, double *dBeta*)

Definition at line 2743 of file cdspop.cc.

References dsp_filter_t.

6.2.1.401 int dsp_filter_init_lpf (dsp_filter_t *dspfilterInst*, float *fPassBand*, float *fStopBand*, float *fRippleRatio*, float *fOverlap*)

clFilter::InitializeLP

Definition at line 2752 of file cdspop.cc.

References dsp_filter_t.

6.2.1.402 int dsp_filter_init_lp (dsp_filter_t *dspfilterInst*, double *dPassBand*, double *dStopBand*, double *dRippleRatio*, double *dOverlap*)

Definition at line 2761 of file cdspop.cc.

References dsp_filter_t.

6.2.1.403 int dsp_filter_init_hpf (dsp_filter_t *dspfilterInst*, float *fPassBand*, float *fStopBand*, float *fRippleRatio*, float *fOverlap*)

clFilter::InitializeHP

Definition at line 2770 of file cdspop.cc.

References dsp_filter_t.

6.2.1.404 int dsp_filter_init_hp (dsp_filter_t *dspfilterInst*, double *dPassBand*, double *dStopBand*, double *dRippleRatio*, double *dOverlap*)

Definition at line 2779 of file cdspop.cc.

References dsp_filter_t.

6.2.1.405 void dsp_filter_set_coeffsf (dsp_filter_t *dspfilterInst*, const float * *fpFiltCoeffs*)

clFilter::SetCoeffs

Definition at line 2788 of file cdspop.cc.

References dsp_filter_t.

6.2.1.406 void dsp_filter_set_coeffs (dsp_filter_t *dspfilterInst*, const double * *dpFiltCoeffs*)

Definition at line 2795 of file cdspop.cc.

References dsp_filter_t.

6.2.1.407 void dsp_filter_set_ccoeffsf (dsp_filter_t *dspfilterInst*, const stpSCplx *spFiltCoeffs*, int *iSmooth*)

Definition at line 2802 of file cdspop.cc.

References dsp_filter_t, and stpSCplx.

6.2.1.408 void dsp_filter_set_ccoeffs (dsp_filter_t *dspfilterInst*, const stpDCplx *spFiltCoeffs*, int *iSmooth*)

Definition at line 2810 of file cdspop.cc.

References dsp_filter_t, and stpDCplx.

6.2.1.409 void dsp_filter_get_coeffsf (dsp_filter_t *dspfilterInst*, float * *fpFiltCoeffs*)

clFilter::GetCoeffs

Definition at line 2818 of file cdspop.cc.

References dsp_filter_t.

6.2.1.410 void dsp_filter_get_coeffs (dsp_filter_t *dspfilterInst*, double ∗ *dpFiltCoeffs*)

Definition at line 2825 of file cdspop.cc.

References dsp_filter_t.

6.2.1.411 void dsp_filter_get_ccoeffsf (dsp_filter_t *dspfilterInst*, stpSCplx *spFiltCoeffs*)

Definition at line 2832 of file cdspop.cc.

References dsp_filter_t, and stpSCplx.

6.2.1.412 void dsp_filter_get_ccoeffs (dsp_filter_t *dspfilterInst*, stpDCplx *spFiltCoeffs*)

Definition at line 2839 of file cdspop.cc.

References dsp_filter_t, and stpDCplx.

6.2.1.413 void dsp_filter_putf (dsp_filter_t *dspfilterInst*, const float ∗ *fpSrcData*, long *lSrcCount*)

clFilter::Put

Definition at line 2846 of file cdspop.cc.

References dsp_filter_t.

6.2.1.414 void dsp_filter_put (dsp_filter_t *dspfilterInst*, const double ∗ *dpSrcData*, long *lSrcCount*)

Definition at line 2853 of file cdspop.cc.

References dsp_filter_t.

6.2.1.415 void dsp_filter_put2f (dsp_filter_t *dspfilterInst*, const float ∗ *fpSrcData*, long *lSrcCount*, const stpSCplx *spCoeffs*)

Definition at line 2860 of file cdspop.cc.

References dsp_filter_t, and stpSCplx.

6.2.1.416 void dsp_filter_put2 (dsp_filter_t *dspfilterInst*, const double * *dpSrcData*, long *lSrcCount*, const stpDCplx *spCoeffs*)

Definition at line 2867 of file cdspop.cc.

References dsp_filter_t, and stpDCplx.

6.2.1.417 void dsp_filter_getf (dsp_filter_t *dspfilterInst*, float * *fpDestData*, long *lDestCount*)

clFilter::Get

Definition at line 2874 of file cdspop.cc.

References dsp_filter_t.

6.2.1.418 void dsp_filter_get (dsp_filter_t *dspfilterInst*, double * *dpDestData*, long *lDestCount*)

Definition at line 2881 of file cdspop.cc.

References dsp_filter_t.

6.2.1.419 void dsp_filter_design_lpf (dsp_filter_t *dspfilterInst*, float * *fpCorner*, int *iDCBlock*)

clFilter::DesignLP

Definition at line 2888 of file cdspop.cc.

References dsp_filter_t.

6.2.1.420 void dsp_filter_design_lp (dsp_filter_t *dspfilterInst*, double * *dpCorner*, int *iDCBlock*)

Definition at line 2896 of file cdspop.cc.

References dsp_filter_t.

6.2.1.421 void dsp_filter_design_lp2f (dsp_filter_t *dspfilterInst*, float * *fpCorner*, float *fSampleRate*, int *iDCBlock*)

Definition at line 2904 of file cdspop.cc.

References dsp_filter_t.

6.2.1.422 void dsp_filter_design_lp2 (dsp_filter_t *dspfilterInst*, double * *dpCorner*, double *dSampleRate*, int *iDCBlock*)

Definition at line 2912 of file cdspop.cc.

References dsp_filter_t.

6.2.1.423 void dsp_filter_design_hpf (dsp_filter_t *dspfilterInst*, float ∗ *fpCorner*)

clFilter::DesignHP

Definition at line 2920 of file cdspop.cc.

References dsp_filter_t.

6.2.1.424 void dsp_filter_design_hp (dsp_filter_t *dspfilterInst*, double ∗ *dpCorner*)

Definition at line 2926 of file cdspop.cc.

References dsp_filter_t.

6.2.1.425 void dsp_filter_design_hp2f (dsp_filter_t *dspfilterInst*, float ∗ *fpCorner*, float *fSampleRate*)

Definition at line 2932 of file cdspop.cc.

References dsp_filter_t.

6.2.1.426 void dsp_filter_design_hp2 (dsp_filter_t *dspfilterInst*, double ∗ *dpCorner*, double *dSampleRate*)

Definition at line 2939 of file cdspop.cc.

References dsp_filter_t.

6.2.1.427 void dsp_filter_design_bpf (dsp_filter_t *dspfilterInst*, float ∗ *fpLowCorner*, float ∗ *fpHighCorner*)

clFilter::DesignBP

Definition at line 2946 of file cdspop.cc.

References dsp_filter_t.

6.2.1.428 void dsp_filter_design_bp (dsp_filter_t *dspfilterInst*, double ∗ *dpLowCorner*, double ∗ *dpHighCorner*)

Definition at line 2953 of file cdspop.cc.

References dsp_filter_t.

6.2.1.429 void dsp_filter_design_bp2f (dsp_filter_t *dspfilterInst*, float ∗ *fpLowCorner*, float ∗ *fpHighCorner*, float *fSampleRate*)

Definition at line 2960 of file cdspop.cc.

References dsp_filter_t.

6.2.1.430 void dsp_filter_design_bp2 (dsp_filter_t *dspfilterInst*, double ∗ *dpLowCorner*, double ∗ *dpHighCorner*, double *dSampleRate*)

Definition at line 2968 of file cdspop.cc.

References dsp_filter_t.

6.2.1.431 void dsp_filter_design_brf (dsp_filter_t *dspfilterInst*, float ∗ *fpLowCorner*, float ∗ *fpHighCorner*)

clFilter::DesignBR

Definition at line 2976 of file cdspop.cc.

References dsp_filter_t.

6.2.1.432 void dsp_filter_design_br (dsp_filter_t *dspfilterInst*, double ∗ *dpLowCorner*, double ∗ *dpHighCorner*)

Definition at line 2983 of file cdspop.cc.

References dsp_filter_t.

6.2.1.433 void dsp_filter_design_br2f (dsp_filter_t *dspfilterInst*, float ∗ *fpLowCorner*, float ∗ *fpHighCorner*, float *fSampleRate*)

Definition at line 2990 of file cdspop.cc.

References dsp_filter_t.

6.2.1.434 void dsp_filter_design_br2 (dsp_filter_t *dspfilterInst*, double ∗ *dpLowCorner*, double ∗ *dpHighCorner*, double *dSampleRate*)

Definition at line 2998 of file cdspop.cc.

References dsp_filter_t.

6.2.1.435 dsp_flip_t dsp_flip_new (void)

Creates new instance of band flipping object.

Returns:
FlipBand instance

Definition at line 3009 of file cdspop.cc.

References dsp_flip_t.

6.2.1.436 void dsp_flip_delete (dsp_flip_t)

Deletes instance of band flipping object.

Parameters:
dspflipInst FlipBand instance

Definition at line 3015 of file cdspop.cc.

References dsp_flip_t.

6.2.1.437 void dsp_flip_initf (dsp_flip_t *dspflipInst*, long *lSize*)

clFlipBand::Initialize

Definition at line 3021 of file cdspop.cc.

References dsp_flip_t.

6.2.1.438 void dsp_flip_init (dsp_flip_t *dspflipInst*, long *lSize*)

Definition at line 3029 of file cdspop.cc.

References dsp_flip_t.

6.2.1.439 void dsp_flip_uninit (dsp_flip_t *dspflipInst*)

clFlipBand::Uninitialize

Definition at line 3037 of file cdspop.cc.

References dsp_flip_t.

6.2.1.440 void dsp_flip_putf (dsp_flip_t *dspflipInst*, const float ∗ *fpSrc*, long *lCount*)

clFlipBand::Put

Definition at line 3043 of file cdspop.cc.

References dsp_flip_t.

6.2.1.441 void dsp_flip_put (dsp_flip_t *dspflipInst*, const double * *dpSrc*, long *lCount*)

Definition at line 3049 of file cdspop.cc.

References dsp_flip_t.

6.2.1.442 void dsp_flip_getf (dsp_flip_t *dspflipInst*, float * *fpDest*, long *lCount*)

clFlipBand::Get

Definition at line 3055 of file cdspop.cc.

References dsp_flip_t.

6.2.1.443 void dsp_flip_get (dsp_flip_t *dspflipInst*, double * *dpDest*, long *lCount*)

Definition at line 3061 of file cdspop.cc.

References dsp_flip_t.

6.2.1.444 void dsp_flip_clear (dsp_flip_t *dspflipInst*)

clFlipBand::Clear

Definition at line 3067 of file cdspop.cc.

References dsp_flip_t.

6.2.1.445 dsp_rebuf_t dsp_rebuf_new (void)

Creates new instance of rebuffering object.

Returns:

Rebuffer instance

Definition at line 3076 of file cdspop.cc.

References dsp_rebuf_t.

6.2.1.446 void dsp_rebuf_delete (dsp_rebuf_t)

Deletes instance of rebuffering object.

Parameters:

dsprebufInst Rebuffer instance

Definition at line 3082 of file cdspop.cc.

References dsp_rebuf_t.

6.2.1.447 void dsp_rebuf_putf (dsp_rebuf_t *dsprebufInst*, const float * *fpSrcData*, long *lSrcCount*)

clReBuffer::Put

Definition at line 3088 of file cdspop.cc.

References dsp_rebuf_t.

6.2.1.448 void dsp_rebuf_put (dsp_rebuf_t *dsprebufInst*, const double * *dpSrcData*, long *lSrcCount*)

Definition at line 3095 of file cdspop.cc.

References dsp_rebuf_t.

6.2.1.449 int dsp_rebuf_getf (dsp_rebuf_t *dsprebufInst*, float * *fpDestData*, long *lDestCount*)

clReBuffer::Get

Definition at line 3102 of file cdspop.cc.

References dsp_rebuf_t.

6.2.1.450 int dsp_rebuf_get (dsp_rebuf_t *dsprebufInst*, double * *dpDestData*, long *lDestCount*)

Definition at line 3111 of file cdspop.cc.

References dsp_rebuf_t.

6.2.1.451 long dsp_rebuf_size (dsp_rebuf_t *dsprebufInst*)

clReBuffer::GetCount

Definition at line 3120 of file cdspop.cc.

References dsp_rebuf_t.

6.2.1.452 void dsp_rebuf_clear (dsp_rebuf_t *dsprebufInst*)

clReBuffer::Clear

Definition at line 3126 of file cdspop.cc.

References dsp_rebuf_t.

6.2.1.453 void dsp_rebuf_copy (dsp_rebuf_t *dsprebufInst*, dsp_rebuf_t *dsprebufCopySrc*)

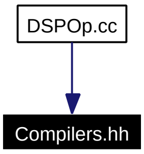
clReBuffer::operator=

Definition at line 3132 of file cdspop.cc.

References dsp_rebuf_t.

6.3 Compilers.hh File Reference

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



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)

Definition at line 31 of file Compilers.hh.

Referenced by clDSPOp::Copy().

6.3.1.2 #define unlikely(x) (x)

Definition at line 32 of file Compilers.hh.

Referenced by clDSPOp::Resample(), and clDSPOp::ResampleAvg().

6.3.1.3 #define prefetch(x, w, l)

Definition at line 33 of file Compilers.hh.

6.4 Condition.hh File Reference

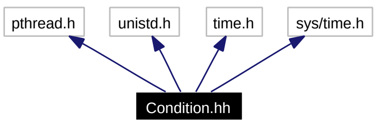
`#include <pthread.h>`

`#include <unistd.h>`

`#include <time.h>`

`#include <sys/time.h>`

Include dependency graph for Condition.hh:



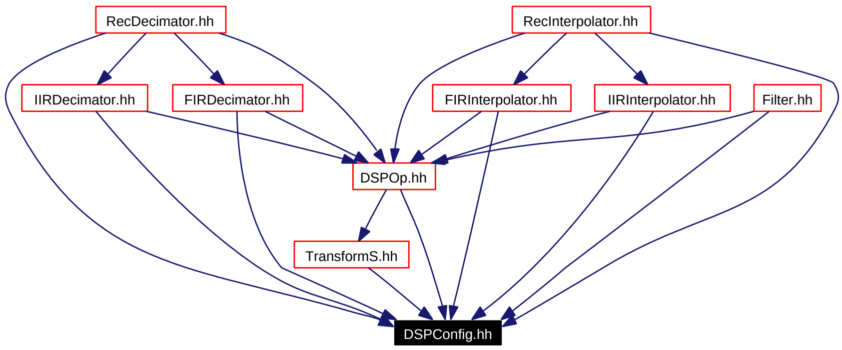
Classes

- class clCondition

  Class implementation of POSIX condition variable.

6.5 DSPConfig.hh File Reference

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



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

Definition at line 47 of file DSPConfig.hh.

Referenced by clRecInterpolator::Initialize(), and clRecDecimator::Initialize().

6.5.1.2 #define DSP_FILT_DEF_OVERLAP 0.75

Definition at line 48 of file DSPConfig.hh.

Referenced by clRecInterpolator::Initialize(), and clRecDecimator::Initialize().

6.5.1.3 #define DSP_FILT_DEF_BETAF 14.96454265f

Definition at line 49 of file DSPConfig.hh.

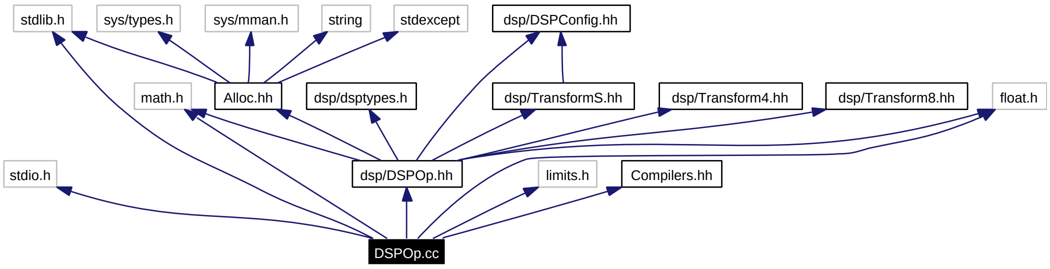
6.5.1.4 #define DSP_FILT_DEF_BETA 14.96454265

Definition at line 50 of file DSPConfig.hh.

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"
```

Include dependency graph for DSPOp.cc:



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*)

Definition at line 59 of file DSPOp.cc.

References INLINE.

Referenced by clDSPOp::Sort().

6.6.1.2 INLINE int DoubleCompare (const void * *vpValue1*, const void * *vpValue2*)

Definition at line 76 of file DSPOp.cc.

References INLINE.

Referenced by clDSPOp::Sort().

6.6.1.3 INLINE int LongCompare (const void * *vpValue1*, const void * *vpValue2*)

Definition at line 93 of file DSPOp.cc.

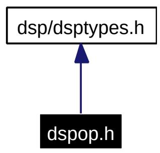
References INLINE.

Referenced by clDSPOp::Sort().

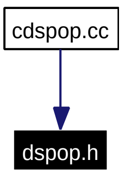
6.7 dspop.h File Reference

`#include "dsp/dsptypes.h"`

Include dependency graph for dspop.h:



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



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_flip_t

    Band flipping 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_zerof (float ∗, long)

  clDSPOp::Zero

- void dsp_zero (double ∗, long)
- void dsp_czerof (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_clipzerof (float ∗, long)

  clDSPOp::ClipZero

- void dsp_clipzero (double ∗, long)
- void dsp_clipzerof_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_autocorrf (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_convertd64f (float ∗, const double ∗, long)
- void dsp_convertf32 (double ∗, const float ∗, long)
- void dsp_convertf32c (unsigned char ∗, const float ∗, long)
- void dsp_convertd64c (unsigned char ∗, const double ∗, long)
- void dsp_convertf32s (signed short ∗, const float ∗, long, int)
- void dsp_convertd64s (signed short ∗, const double ∗, long, int)
- void dsp_convertf32i (signed int ∗, const float ∗, long, int)
- void dsp_convertd64i (signed int ∗, const double ∗, long, int)
- void dsp_cart2polarf (float ∗, float ∗, const float ∗, const float ∗, long)

  clDSPOp::CartToPolar

- void dsp_cart2polar (double ∗, double ∗, const double ∗, const double ∗, long)
- void dsp_cart2polar2f (float ∗, float ∗, const stpSCplx, long)
- void dsp_cart2polar2 (double ∗, double ∗, const stpDCplx, long)
- void dsp_cart2polar3f (stpSPolar, const stpSCplx, long)
- void dsp_cart2polar3 (stpDPolar, const stpDCplx, long)
- void dsp_cart2polar4f (utpSCoord, long)
- void dsp_cart2polar4 (utpDCoord, long)
- void dsp_polar2cartf (float ∗, float ∗, const float ∗, const float ∗, long)

  clDSPOp::PolarToCart

- void dsp_polar2cart (double ∗, double ∗, const double ∗, const double ∗, long)
- void dsp_polar2cart2f (stpSCplx, const float ∗, const float ∗, long)
- void dsp_polar2cart2 (stpDCplx, const double ∗, const double ∗, long)

- void dsp_polar2cart3f (stpSCplx, const stpSPolar, long)
- void dsp_polar2cart3 (stpDCplx, const stpDPolar, long)
- void dsp_polar2cart4f (utpSCoord, long)
- void dsp_polar2cart4 (utpDCoord, long)
- float dsp_crosscorrf (const float ∗, const float ∗, long)

  clDSPOp::CrossCorr

- double dsp_crosscorr (const double ∗, const double ∗, long)
- float dsp_crosscorr2f (const float ∗, const float ∗, long, long)

  clDSPOp::DelCrossCorr

- double dsp_crosscorr2 (const double ∗, const double ∗, long, long)
- void dsp_crosscorr3f (float ∗, const float ∗, const float ∗, long, const long ∗, long)
- void dsp_crosscorr3 (double ∗, const double ∗, const double ∗, long, const long ∗, long)
- float dsp_energyf (const float ∗, long)

  clDSPOp::Energy

- double dsp_energy (const double ∗, long)
- void dsp_magnitudef (float ∗, const stpSCplx, long)

  clDSPOp::Magnitude

- void dsp_magnitude (double ∗, const stpDCplx, long)
- 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_convertd2cf (stpSCplx, const double ∗, long)

- void dsp_fftw_convertd2cd (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_bartlettf (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)
- void dsp_dct_init (dsp_t, long)

  clDSPOp::DCTInitialize

- void dsp_dct_uninit (dsp_t)

  clDSPOp::DCTUninitialize

- void dsp_dctf (dsp_t, float ∗)

clDSPOp::DCTi

- void dsp_dct (dsp_t, double ∗)
- void dsp_dctf_nip (dsp_t, float ∗, const float ∗)

  clDSPOp::DCTo

- void dsp_dct_nip (dsp_t, double ∗, const double ∗)
- void dsp_idctf (dsp_t, float ∗)

  clDSPOp::IDCTi

- void dsp_idct (dsp_t, double ∗)
- void dsp_idctf_nip (dsp_t, float ∗, const float ∗)

  clDSPOp::IDCTo

- void dsp_idct_nip (dsp_t, double ∗, const double ∗)
- 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_decfft_t, const double ∗, long)
- int dsp_dec_fft_getf (dsp_decfft_t, float ∗, long)

  clFFTDecimator::Get

- int dsp_dec_fft_get (dsp_decfft_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_coeffsf (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_coeffsf (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_brf (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_flip_t dsp_flip_new (void)

  Creates new instance of band flipping object.

- void dsp_flip_delete (dsp_flip_t)

  Deletes instance of band flipping object.

- void dsp_flip_initf (dsp_flip_t, long)

  clFlipBand::Initialize

- void dsp_flip_init (dsp_flip_t, long)
- void dsp_flip_uninit (dsp_flip_t)

clFlipBand::Uninitialize

- void dsp_flip_putf (dsp_flip_t, const float ∗, long)

  clFlipBand::Put

- void dsp_flip_put (dsp_flip_t, const double ∗, long)
- void dsp_flip_getf (dsp_flip_t, float ∗, long)

  clFlipBand::Get

- void dsp_flip_get (dsp_flip_t, double ∗, long)
- void dsp_flip_clear (dsp_flip_t)

  clFlipBand::Clear

- 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.

Definition at line 35 of file dspop.h.

Referenced by dsp_cfft_nip(), dsp_cfftf_nip(), dsp_cifft_nip(), dsp_cifftf_nip(), dsp_dct(), dsp_dct_init(), dsp_dct_nip(), dsp_dct_uninit(), dsp_dctf(), dsp_dctf_nip(), dsp_deg2rad(), dsp_deg2radf(), dsp_delete(), dsp_fft(), dsp_fft_init(), dsp_fft_nip(), dsp_fft_uninit(), dsp_fftf(), dsp_fftf_nip(), dsp_fir_allocate(), dsp_fir_allocatef(), dsp_fir_filter(), dsp_fir_filter_fst(), dsp_fir_filter_nip(), dsp_fir_filterf(), dsp_fir_filterf_fst(), dsp_fir_filterf_nip(), dsp_fir_free(), dsp_idct(), dsp_idct_nip(), dsp_idctf(), dsp_idctf_nip(), dsp_ifft_nip(), dsp_ifftf_nip(), dsp_iir_clear(), dsp_iir_filter(), dsp_iir_filter_nip(), dsp_iir_filterf(), dsp_iir_filterf_nip(), dsp_iir_init(), dsp_iir_initf(), dsp_new(), dsp_rad2deg(), dsp_rad2degf(), dsp_rebuffer(), dsp_rebufferf(), dsp_win_bartlett(), dsp_win_bartlettf(), dsp_win_blackman(), dsp_win_blackman_harris(), dsp_win_blackman_harrisf(), dsp_win_blackmanf(), dsp_win_cos_tapered(), dsp_win_cos_taperedf(), dsp_win_dolph_chebyshev(), dsp_win_dolph_chebyshevf(), dsp_win_exact_blackman(), dsp_win_exact_blackmanf(), dsp_win_flat_top(), dsp_win_flat_topf(), dsp_win_generic_cos(), dsp_win_generic_cosf(), dsp_win_hamming(), dsp_win_hammingf(), dsp_win_hanning(), dsp_win_hanningf(), dsp_win_kaiser(), dsp_win_kaiser_bessel(), dsp_win_kaiser_besself(), dsp_win_kaiserf(), dsp_win_tukey(), and dsp_win_tukeyf().

6.7.1.2 typedef void* dsp_iircas_t

Cascaded IIR filter object type.

Definition at line 37 of file dspop.h.

Referenced by dsp_iir_cas_clear(), dsp_iir_cas_delete(), dsp_iir_cas_init(), dsp_iir_cas_initf(), dsp_iir_cas_new(), dsp_iir_cas_process(), dsp_iir_cas_process_nip(), dsp_iir_cas_processf(), dsp_iir_cas_processf_nip(), and dsp_iir_cas_uninit().

6.7.1.3 typedef void* dsp_decfft_t

FFT decimator object type.

Definition at line 39 of file dspop.h.

Referenced by dsp_dec_fft_delete(), dsp_dec_fft_get(), dsp_dec_fft_getf(), dsp_dec_fft_init(), dsp_dec_fft_initf(), dsp_dec_fft_new(), dsp_dec_fft_put(), dsp_dec_fft_putf(), and dsp_dec_fft_uninit().

6.7.1.4 typedef void* dsp_decfir_t

FIR decimator object type.

Definition at line 41 of file dspop.h.

Referenced by dsp_dec_fir_delete(), dsp_dec_fir_get(), dsp_dec_fir_getf(), dsp_dec_fir_init(), dsp_dec_fir_initf(), dsp_dec_fir_new(), dsp_dec_fir_put(), dsp_dec_fir_-

putf(), and dsp_dec_fir_uninit().

6.7.1.5 typedef void* dsp_deciir_t

IIR decimator object type.

Definition at line 43 of file dspop.h.

Referenced by dsp_dec_iir_delete(), dsp_dec_iir_get(), dsp_dec_iir_getf(), dsp_dec_iir_init(), dsp_dec_iir_initf(), dsp_dec_iir_new(), dsp_dec_iir_put(), dsp_dec_iir_putf(), and dsp_dec_iir_uninit().

6.7.1.6 typedef void* dsp_decrec_t

Recursive decimator object type.

Definition at line 45 of file dspop.h.

Referenced by dsp_dec_rec_delete(), dsp_dec_rec_get(), dsp_dec_rec_getf(), dsp_dec_rec_init(), dsp_dec_rec_initf(), dsp_dec_rec_new(), dsp_dec_rec_put(), dsp_dec_rec_putf(), and dsp_dec_rec_uninit().

6.7.1.7 typedef void* dsp_intfft_t

FFT interpolator object type.

Definition at line 47 of file dspop.h.

Referenced by dsp_int_fft_delete(), dsp_int_fft_get(), dsp_int_fft_getf(), dsp_int_fft_init(), dsp_int_fft_initf(), dsp_int_fft_new(), dsp_int_fft_put(), dsp_int_fft_putf(), and dsp_int_fft_uninit().

6.7.1.8 typedef void* dsp_intfir_t

FIR interpolator object type.

Definition at line 49 of file dspop.h.

Referenced by dsp_int_fir_delete(), dsp_int_fir_get(), dsp_int_fir_getf(), dsp_int_fir_init(), dsp_int_fir_initf(), dsp_int_fir_new(), dsp_int_fir_put(), dsp_int_fir_putf(), and dsp_int_fir_uninit().

6.7.1.9 typedef void* dsp_intiir_t

IIR interpolator object type.

Definition at line 51 of file dspop.h.

Referenced by dsp_int_iir_delete(), dsp_int_iir_get(), dsp_int_iir_getf(), dsp_int_iir_init(), dsp_int_iir_initf(), dsp_int_iir_new(), dsp_int_iir_put(), dsp_int_iir_putf(), and dsp_int_iir_uninit().

6.7.1.10 typedef void* dsp_intrec_t

Recursive interpolator object type.

Definition at line 53 of file dspop.h.

Referenced by dsp_int_rec_delete(), dsp_int_rec_get(), dsp_int_rec_getf(), dsp_int_rec_init(), dsp_int_rec_initf(), dsp_int_rec_new(), dsp_int_rec_put(), dsp_int_rec_putf(), and dsp_int_rec_uninit().

6.7.1.11 typedef void* dsp_filter_t

FFT filter object type.

Definition at line 55 of file dspop.h.

Referenced by dsp_filter_delete(), dsp_filter_design_bp(), dsp_filter_design_bp2(), dsp_filter_design_bp2f(), dsp_filter_design_bpf(), dsp_filter_design_br(), dsp_filter_design_br2(), dsp_filter_design_br2f(), dsp_filter_design_brf(), dsp_filter_design_hp(), dsp_filter_design_hp2(), dsp_filter_design_hp2f(), dsp_filter_design_hpf(), dsp_filter_design_lp(), dsp_filter_design_lp2(), dsp_filter_design_lp2f(), dsp_filter_design_lpf(), dsp_filter_get(), dsp_filter_get_ccoeffs(), dsp_filter_get_ccoeffsf(), dsp_filter_get_coeffs(), dsp_filter_get_coeffsf(), dsp_filter_getf(), dsp_filter_init(), dsp_filter_init2(), dsp_filter_init2f(), dsp_filter_init_hp(), dsp_filter_init_hpf(), dsp_filter_init_lp(), dsp_filter_init_lpf(), dsp_filter_initf(), dsp_filter_new(), dsp_filter_put(), dsp_filter_put2(), dsp_filter_put2f(), dsp_filter_putf(), dsp_filter_set_ccoeffs(), dsp_filter_set_ccoeffsf(), dsp_filter_set_coeffs(), and dsp_filter_set_coeffsf().

6.7.1.12 typedef void* dsp_flip_t

Band flipping object type.

Definition at line 57 of file dspop.h.

Referenced by dsp_flip_clear(), dsp_flip_delete(), dsp_flip_get(), dsp_flip_getf(), dsp_flip_init(), dsp_flip_initf(), dsp_flip_new(), dsp_flip_put(), dsp_flip_putf(), and dsp_flip_uninit().

6.7.1.13 typedef void* dsp_rebuf_t

Rebuffering object type.

Definition at line 59 of file dspop.h.

Referenced by dsp_rebuf_clear(), dsp_rebuf_copy(), dsp_rebuf_delete(), dsp_rebuf_get(), dsp_rebuf_getf(), dsp_rebuf_new(), dsp_rebuf_put(), dsp_rebuf_putf(), and dsp_rebuf_size().

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.

Definition at line 53 of file cdspop.cc.

6.7.2.2 signed long dsp_roundf (float)

clDSPOp::Round

Definition at line 62 of file cdspop.cc.

References clDSPOp::Round().

6.7.2.3 signed long dsp_round (double)

Definition at line 68 of file cdspop.cc.

References clDSPOp::Round().

6.7.2.4 void dsp_addf (float ∗, float, long)

clDSPOp::Add

Definition at line 74 of file cdspop.cc.

References clDSPOp::Add().

6.7.2.5 void dsp_add (double ∗, double, long)

Definition at line 80 of file cdspop.cc.

References clDSPOp::Add().

6.7.2.6 void dsp_caddf (stpSCplx, stSCplx, long)

Definition at line 86 of file cdspop.cc.

References clDSPOp::Add(), stpSCplx, and stSCplx.

6.7.2.7 void dsp_cadd (stpDCplx, stDCplx, long)

Definition at line 92 of file cdspop.cc.

References clDSPOp::Add(), stDCplx, and stpDCplx.

6.7.2.8 void dsp_add2f (float $*$, const float $*$, long)

Definition at line 98 of file cdspop.cc.

References clDSPOp::Add().

6.7.2.9 void dsp_add2 (double $*$, const double $*$, long)

Definition at line 104 of file cdspop.cc.

References clDSPOp::Add().

6.7.2.10 void dsp_cadd2f (stpSCplx, const *stpSCplx*, long)

Definition at line 110 of file cdspop.cc.

References clDSPOp::Add(), and stpSCplx.

6.7.2.11 void dsp_cadd2 (stpDCplx, const *stpDCplx*, long)

Definition at line 116 of file cdspop.cc.

References clDSPOp::Add(), and stpDCplx.

6.7.2.12 void dsp_add3f (float $*$, const float $*$, const float $*$, long)

Definition at line 122 of file cdspop.cc.

References clDSPOp::Add().

6.7.2.13 void dsp_add3 (double $*$, const double $*$, const double $*$, long)

Definition at line 129 of file cdspop.cc.

References clDSPOp::Add().

6.7.2.14 void dsp_cadd3f (stpSCplx, const *stpSCplx*, const *stpSCplx*, long)

Definition at line 136 of file cdspop.cc.

References clDSPOp::Add(), and stpSCplx.

6.7.2.15 void dsp_cadd3 (stpDCplx, const *stpDCplx*, const *stpDCplx*, long)

Definition at line 143 of file cdspop.cc.

References clDSPOp::Add(), and stpDCplx.

6.7.2.16 void dsp_subf (float *, float, long)

clDSPOp::Sub

Definition at line 150 of file cdspop.cc.

References clDSPOp::Sub().

6.7.2.17 void dsp_sub (double *, double, long)

Definition at line 156 of file cdspop.cc.

References clDSPOp::Sub().

6.7.2.18 void dsp_csubf (stpSCplx, stSCplx, long)

Definition at line 162 of file cdspop.cc.

References stpSCplx, stSCplx, and clDSPOp::Sub().

6.7.2.19 void dsp_csub (stpDCplx, stDCplx, long)

Definition at line 168 of file cdspop.cc.

References stDCplx, stpDCplx, and clDSPOp::Sub().

6.7.2.20 void dsp_sub2f (float *, const float *, long)

Definition at line 174 of file cdspop.cc.

References clDSPOp::Sub().

6.7.2.21 void dsp_sub2 (double *, const double *, long)

Definition at line 180 of file cdspop.cc.

References clDSPOp::Sub().

6.7.2.22 void dsp_csub2f (stpSCplx, const *stpSCplx*, long)

Definition at line 186 of file cdspop.cc.

References stpSCplx, and clDSPOp::Sub().

6.7.2.23 void dsp_csub2 (stpDCplx, const *stpDCplx*, long)

Definition at line 192 of file cdspop.cc.

References stpDCplx, and clDSPOp::Sub().

6.7.2.24 void dsp_sub3f (float *, const float *, const float *, long)

Definition at line 198 of file cdspop.cc.

References clDSPOp::Sub().

6.7.2.25 void dsp_sub3 (double *, const double *, const double *, long)

Definition at line 205 of file cdspop.cc.

References clDSPOp::Sub().

6.7.2.26 void dsp_csub3f (stpSCplx, const *stpSCplx*, const *stpSCplx*, long)

Definition at line 212 of file cdspop.cc.

References stpSCplx, and clDSPOp::Sub().

6.7.2.27 void dsp_csub3 (stpDCplx, const *stpDCplx*, const *stpDCplx*, long)

Definition at line 219 of file cdspop.cc.

References stpDCplx, and clDSPOp::Sub().

6.7.2.28 void dsp_mulf (float *, float, long)

clDSPOp::Mul

Definition at line 226 of file cdspop.cc.

References clDSPOp::Mul().

6.7.2.29 void dsp_mul (double *, double, long)

Definition at line 232 of file cdspop.cc.

References clDSPOp::Mul().

6.7.2.30 void dsp_chmulf (stpSCplx, float, long)

Definition at line 238 of file cdspop.cc.

References clDSPOp::Mul(), and stpSCplx.

6.7.2.31 void dsp_chmul (stpDCplx, double, long)

Definition at line 244 of file cdspop.cc.

References clDSPOp::Mul(), and stpDCplx.

6.7.2.32 void dsp_cmulf (stpSCplx, stSCplx, long)

Definition at line 250 of file cdspop.cc.

References clDSPOp::Mul(), stpSCplx, and stSCplx.

6.7.2.33 void dsp_cmul (stpDCplx, stDCplx, long)

Definition at line 256 of file cdspop.cc.

References clDSPOp::Mul(), stDCplx, and stpDCplx.

6.7.2.34 void dsp_mulf_nip (float *, const float *, float, long)

Definition at line 262 of file cdspop.cc.

References clDSPOp::Mul().

6.7.2.35 void dsp_mul_nip (double *, const double *, double, long)

Definition at line 269 of file cdspop.cc.

References clDSPOp::Mul().

6.7.2.36 void dsp_mul2f (float *, const float *, long)

Definition at line 276 of file cdspop.cc.

References clDSPOp::Mul().

6.7.2.37 void dsp_mul2 (double *, const double *, long)

Definition at line 282 of file cdspop.cc.

References clDSPOp::Mul().

6.7.2.38 void dsp_chmul2f (stpSCplx, const float *, long)

Definition at line 288 of file cdspop.cc.

References clDSPOp::Mul(), and stpSCplx.

6.7.2.39 void dsp_chmul2 (stpDCplx, const double *, long)

Definition at line 294 of file cdspop.cc.

References clDSPOp::Mul(), and stpDCplx.

6.7.2.40 void dsp_cmul2f (stpSCplx, const *stpSCplx*, long)

Definition at line 300 of file cdspop.cc.

References clDSPOp::Mul(), and stpSCplx.

6.7.2.41 void dsp_cmul2 (stpDCplx, const *stpDCplx*, long)

Definition at line 306 of file cdspop.cc.

References clDSPOp::Mul(), and stpDCplx.

6.7.2.42 void dsp_mul3f (float ∗, const float ∗, const float ∗, long)

Definition at line 312 of file cdspop.cc.

References clDSPOp::Mul().

6.7.2.43 void dsp_mul3 (double ∗, const double ∗, const double ∗, long)

Definition at line 319 of file cdspop.cc.

References clDSPOp::Mul().

6.7.2.44 void dsp_cmul3f (stpSCplx, const *stpSCplx*, const *stpSCplx*, long)

Definition at line 326 of file cdspop.cc.

References clDSPOp::Mul(), and stpSCplx.

6.7.2.45 void dsp_cmul3 (stpDCplx, const *stpDCplx*, const *stpDCplx*, long)

Definition at line 333 of file cdspop.cc.

References clDSPOp::Mul(), and stpDCplx.

6.7.2.46 void dsp_ccmulf (stpSCplx, const *stpSCplx*, long)

Definition at line 340 of file cdspop.cc.

References clDSPOp::MulC(), and stpSCplx.

6.7.2.47 void dsp_ccmul (stpDCplx, const *stpDCplx*, long)

Definition at line 346 of file cdspop.cc.

References clDSPOp::MulC(), and stpDCplx.

6.7.2.48 void dsp_ccmulf_nip (stpSCplx, const *stpSCplx*, const *stpSCplx*, long)

Definition at line 352 of file cdspop.cc.

References clDSPOp::MulC(), and stpSCplx.

6.7.2.49 void dsp_ccmul_nip (stpDCplx, const *stpDCplx*, const *stpDCplx*, long)

Definition at line 359 of file cdspop.cc.

References clDSPOp::MulC(), and stpDCplx.

6.7.2.50 void dsp_divf (float *, float, long)

clDSPOp::Div

Definition at line 366 of file cdspop.cc.

References clDSPOp::Div().

6.7.2.51 void dsp_div (double *, double, long)

Definition at line 372 of file cdspop.cc.

References clDSPOp::Div().

6.7.2.52 void dsp_cdivf (stpSCplx, stSCplx, long)

Definition at line 378 of file cdspop.cc.

References clDSPOp::Div(), stpSCplx, and stSCplx.

6.7.2.53 void dsp_cdiv (stpDCplx, stDCplx, long)

Definition at line 384 of file cdspop.cc.

References clDSPOp::Div(), stDCplx, and stpDCplx.

6.7.2.54 void dsp_div2f (float *, const float *, long)

Definition at line 390 of file cdspop.cc.

References clDSPOp::Div().

6.7.2.55 void dsp_div2 (double *, const double *, long)

Definition at line 396 of file cdspop.cc.

References clDSPOp::Div().

6.7.2.56 void dsp_cdiv2f (stpSCplx, const *stpSCplx*, long)

Definition at line 402 of file cdspop.cc.

References clDSPOp::Div(), and stpSCplx.

6.7.2.57 void dsp_cdiv2 (stpDCplx, const *stpDCplx*, long)

Definition at line 408 of file cdspop.cc.

References clDSPOp::Div(), and stpDCplx.

6.7.2.58 void dsp_div3f (float ∗, const float ∗, const float ∗, long)

Definition at line 414 of file cdspop.cc.

References clDSPOp::Div().

6.7.2.59 void dsp_div3 (double ∗, const double ∗, const double ∗, long)

Definition at line 421 of file cdspop.cc.

References clDSPOp::Div().

6.7.2.60 void dsp_cdiv3f (stpSCplx, const *stpSCplx*, const *stpSCplx*, long)

Definition at line 428 of file cdspop.cc.

References clDSPOp::Div(), and stpSCplx.

6.7.2.61 void dsp_cdiv3 (stpDCplx, const *stpDCplx*, const *stpDCplx*, long)

Definition at line 435 of file cdspop.cc.

References clDSPOp::Div(), and stpDCplx.

6.7.2.62 void dsp_div1xf (float ∗, long)

clDSPOp::Div1x

Definition at line 442 of file cdspop.cc.

References clDSPOp::Div1x().

6.7.2.63 void dsp_div1x (double ∗, long)

Definition at line 448 of file cdspop.cc.

References clDSPOp::Div1x().

6.7.2.64 void dsp_div1xf_nip (float $*$, const float $*$, long)

Definition at line 454 of file cdspop.cc.

References clDSPOp::Div1x().

6.7.2.65 void dsp_div1x_nip (double $*$, const double $*$, long)

Definition at line 460 of file cdspop.cc.

References clDSPOp::Div1x().

6.7.2.66 void dsp_muladdf (float $*$, float, float, long)

clDSPOp::MulAdd

Definition at line 466 of file cdspop.cc.

References clDSPOp::MulAdd().

6.7.2.67 void dsp_muladd (double $*$, double, double, long)

Definition at line 472 of file cdspop.cc.

References clDSPOp::MulAdd().

6.7.2.68 void dsp_muladdf_nip (float $*$, const float $*$, float, float, long)

Definition at line 478 of file cdspop.cc.

References clDSPOp::MulAdd().

6.7.2.69 void dsp_muladd_nip (double $*$, const double $*$, double, double, long)

Definition at line 485 of file cdspop.cc.

References clDSPOp::MulAdd().

6.7.2.70 void dsp_absf (float $*$, long)

clDSPOp::Abs

Definition at line 492 of file cdspop.cc.

References clDSPOp::Abs().

6.7.2.71 void dsp_abs (double $*$, long)

Definition at line 498 of file cdspop.cc.

References clDSPOp::Abs().

6.7.2.72 void dsp_absf_nip (float ∗, const float ∗, long)

Definition at line 504 of file cdspop.cc.

References clDSPOp::Abs().

6.7.2.73 void dsp_abs_nip (double ∗, const double ∗, long)

Definition at line 510 of file cdspop.cc.

References clDSPOp::Abs().

6.7.2.74 void dsp_sqrtf (float ∗, long)

clDSPOp::Sqrt

Definition at line 516 of file cdspop.cc.

References clDSPOp::Sqrt().

6.7.2.75 void dsp_sqrt (double ∗, long)

Definition at line 522 of file cdspop.cc.

References clDSPOp::Sqrt().

6.7.2.76 void dsp_sqrtf_nip (float ∗, const float ∗, long)

Definition at line 528 of file cdspop.cc.

References clDSPOp::Sqrt().

6.7.2.77 void dsp_sqrt_nip (double ∗, const double ∗, long)

Definition at line 534 of file cdspop.cc.

References clDSPOp::Sqrt().

6.7.2.78 void dsp_zerof (float ∗, long)

clDSPOp::Zero

Definition at line 540 of file cdspop.cc.

References clDSPOp::Zero().

6.7.2.79 void dsp_zero (double *, long)

Definition at line 546 of file cdspop.cc.

References clDSPOp::Zero().

6.7.2.80 void dsp_czerof (stpSCplx, long)

Definition at line 552 of file cdspop.cc.

References stpSCplx, and clDSPOp::Zero().

6.7.2.81 void dsp_czero (stpDCplx, long)

Definition at line 558 of file cdspop.cc.

References stpDCplx, and clDSPOp::Zero().

6.7.2.82 void dsp_setf (float *, float, long)

clDSPOp::Set

Definition at line 564 of file cdspop.cc.

References clDSPOp::Set().

6.7.2.83 void dsp_set (double *, double, long)

Definition at line 570 of file cdspop.cc.

References clDSPOp::Set().

6.7.2.84 void dsp_csetf (stpSCplx, stSCplx, long)

Definition at line 576 of file cdspop.cc.

References clDSPOp::Set(), stpSCplx, and stSCplx.

6.7.2.85 void dsp_cset (stpDCplx, stDCplx, long)

Definition at line 582 of file cdspop.cc.

References clDSPOp::Set(), stDCplx, and stpDCplx.

6.7.2.86 void dsp_set2f (float *, float, long, long, long)

Definition at line 588 of file cdspop.cc.

References clDSPOp::Set().

6.7.2.87 void dsp_set2 (double *, double, long, long, long)

Definition at line 595 of file cdspop.cc.

References clDSPOp::Set().

6.7.2.88 void dsp_cset2f (stpSCplx, stSCplx, long, long, long)

Definition at line 602 of file cdspop.cc.

References clDSPOp::Set(), stpSCplx, and stSCplx.

6.7.2.89 void dsp_cset2 (stpDCplx, stDCplx, long, long, long)

Definition at line 609 of file cdspop.cc.

References clDSPOp::Set(), stDCplx, and stpDCplx.

6.7.2.90 void dsp_clipf (float *, float, long)

clDSPOp::Clip

Definition at line 616 of file cdspop.cc.

References clDSPOp::Clip().

6.7.2.91 void dsp_clip (double *, double, long)

Definition at line 622 of file cdspop.cc.

References clDSPOp::Clip().

6.7.2.92 void dsp_clipf_nip (float *, const float *, float, long)

Definition at line 628 of file cdspop.cc.

References clDSPOp::Clip().

6.7.2.93 void dsp_clip_nip (double *, const double *, double, long)

Definition at line 635 of file cdspop.cc.

References clDSPOp::Clip().

6.7.2.94 void dsp_clip2f (float *, float, float, long)

Definition at line 642 of file cdspop.cc.

References clDSPOp::Clip().

6.7.2.95 void dsp_clip2 (double ∗, double, double, long)

Definition at line 648 of file cdspop.cc.

References clDSPOp::Clip().

6.7.2.96 void dsp_clip2f_nip (float ∗, const float ∗, float, float, long)

Definition at line 654 of file cdspop.cc.

References clDSPOp::Clip().

6.7.2.97 void dsp_clip2_nip (double ∗, const double ∗, double, double, long)

Definition at line 661 of file cdspop.cc.

References clDSPOp::Clip().

6.7.2.98 void dsp_clipzerof (float ∗, long)

clDSPOp::ClipZero

Definition at line 668 of file cdspop.cc.

References clDSPOp::ClipZero().

6.7.2.99 void dsp_clipzero (double ∗, long)

Definition at line 674 of file cdspop.cc.

References clDSPOp::ClipZero().

6.7.2.100 void dsp_clipzerof_nip (float ∗, const float ∗, long)

Definition at line 680 of file cdspop.cc.

References clDSPOp::ClipZero().

6.7.2.101 void dsp_clipzero_nip (double ∗, const double ∗, long)

Definition at line 686 of file cdspop.cc.

References clDSPOp::ClipZero().

6.7.2.102 void dsp_copyf (float ∗, const float ∗, long)

clDSPOp::Copy

Definition at line 692 of file cdspop.cc.

References clDSPOp::Copy().

6.7.2.103 void dsp_copy (double $*$, const double $*$, long)

Definition at line 698 of file cdspop.cc.

References clDSPOp::Copy().

6.7.2.104 float dsp_convolvef (const float $*$, const float $*$, long)

clDSPOp::Convolve

Definition at line 716 of file cdspop.cc.

References clDSPOp::Convolve().

6.7.2.105 double dsp_convolve (const double $*$, const double $*$, long)

Definition at line 722 of file cdspop.cc.

References clDSPOp::Convolve().

6.7.2.106 void dsp_convolve2f (float $*$, const float $*$, const float $*$, long)

Definition at line 728 of file cdspop.cc.

References clDSPOp::Convolve().

6.7.2.107 void dsp_convolve2 (double $*$, const double $*$, const double $*$, long)

Definition at line 735 of file cdspop.cc.

References clDSPOp::Convolve().

6.7.2.108 float dsp_correlatef (const float $*$, const float $*$, long)

clDSPOp::Correlate

Definition at line 742 of file cdspop.cc.

References clDSPOp::Correlate().

6.7.2.109 double dsp_correlate (const double $*$, const double $*$, long)

Definition at line 748 of file cdspop.cc.

References clDSPOp::Correlate().

6.7.2.110 void dsp_correlate2f (float ∗, const float ∗, const float ∗, long)

Definition at line 754 of file cdspop.cc.

References clDSPOp::Correlate().

6.7.2.111 void dsp_correlate2 (double ∗, const double ∗, const double ∗, long)

Definition at line 761 of file cdspop.cc.

References clDSPOp::Correlate().

6.7.2.112 float dsp_autocorrf (const float ∗, long)

clDSPOp::AutoCorrelate

Definition at line 768 of file cdspop.cc.

References clDSPOp::AutoCorrelate().

6.7.2.113 double dsp_autocorr (const double ∗, long)

Definition at line 774 of file cdspop.cc.

References clDSPOp::AutoCorrelate().

6.7.2.114 void dsp_autocorr2f (float ∗, const float ∗, long)

Definition at line 780 of file cdspop.cc.

References clDSPOp::AutoCorrelate().

6.7.2.115 void dsp_autocorr2 (double ∗, const double ∗, long)

Definition at line 786 of file cdspop.cc.

References clDSPOp::AutoCorrelate().

6.7.2.116 float dsp_dotproductf (const float ∗, const float ∗, long)

clDSPOp::DotProduct

Definition at line 792 of file cdspop.cc.

References clDSPOp::DotProduct().

6.7.2.117 double dsp_dotproduct (const double ∗, const double ∗, long)

Definition at line 798 of file cdspop.cc.

References clDSPOp::DotProduct().

6.7.2.118 void dsp_minmaxf (float $*$, float $*$, const float $*$, long)

clDSPOp::MinMax

Definition at line 804 of file cdspop.cc.

References clDSPOp::MinMax().

6.7.2.119 void dsp_minmax (double $*$, double $*$, const double $*$, long)

Definition at line 810 of file cdspop.cc.

References clDSPOp::MinMax().

6.7.2.120 float dsp_meanf (const float $*$, long)

clDSPOp::Mean

Definition at line 816 of file cdspop.cc.

References clDSPOp::Mean().

6.7.2.121 double dsp_mean (const double $*$, long)

Definition at line 822 of file cdspop.cc.

References clDSPOp::Mean().

6.7.2.122 float dsp_meadianf (const float $*$, long)

clDSPOp::Median

6.7.2.123 double dsp_median (const double $*$, long)

Definition at line 834 of file cdspop.cc.

References clDSPOp::Median().

6.7.2.124 void dsp_negatef (float $*$, long)

clDSPOp::Negate

Definition at line 840 of file cdspop.cc.

References clDSPOp::Negate().

6.7.2.125 void dsp_negate (double ∗, long)

Definition at line 846 of file cdspop.cc.

References clDSPOp::Negate().

6.7.2.126 void dsp_negatef_nip (float ∗, const float ∗, long)

Definition at line 852 of file cdspop.cc.

References clDSPOp::Negate().

6.7.2.127 void dsp_negate_nip (double ∗, const double ∗, long)

Definition at line 858 of file cdspop.cc.

References clDSPOp::Negate().

6.7.2.128 void dsp_normalizef (float ∗, long)

clDSPOp::Normalize

Definition at line 864 of file cdspop.cc.

References clDSPOp::Normalize().

6.7.2.129 void dsp_normalize (double ∗, long)

Definition at line 870 of file cdspop.cc.

References clDSPOp::Normalize().

6.7.2.130 void dsp_normalizef_nip (float ∗, const float ∗, long)

Definition at line 876 of file cdspop.cc.

References clDSPOp::Normalize().

6.7.2.131 void dsp_normalize_nip (double ∗, const double ∗, long)

Definition at line 882 of file cdspop.cc.

References clDSPOp::Normalize().

6.7.2.132 float dsp_productf (const float ∗, long)

clDSPOp::Product

Definition at line 888 of file cdspop.cc.

References clDSPOp::Product().

6.7.2.133 double dsp_product (const double $*$, long)

Definition at line 894 of file cdspop.cc.

References clDSPOp::Product().

6.7.2.134 void dsp_reversef (float $*$, long)

clDSPOp::Reverse

Definition at line 900 of file cdspop.cc.

References clDSPOp::Reverse().

6.7.2.135 void dsp_reverse (double $*$, long)

Definition at line 906 of file cdspop.cc.

References clDSPOp::Reverse().

6.7.2.136 void dsp_reversef_nip (float $*$, const float $*$, long)

Definition at line 912 of file cdspop.cc.

References clDSPOp::Reverse().

6.7.2.137 void dsp_reverse_nip (double $*$, const double $*$, long)

Definition at line 918 of file cdspop.cc.

References clDSPOp::Reverse().

6.7.2.138 void dsp_scalef (float $*$, long)

clDSPOp::Scale

Definition at line 924 of file cdspop.cc.

References clDSPOp::Scale().

6.7.2.139 void dsp_scale (double $*$, long)

Definition at line 930 of file cdspop.cc.

References clDSPOp::Scale().

6.7.2.140 void dsp_scalef_nip (float ∗, const float ∗, long)

Definition at line 936 of file cdspop.cc.

References clDSPOp::Scale().

6.7.2.141 void dsp_scale_nip (double ∗, const double ∗, long)

Definition at line 942 of file cdspop.cc.

References clDSPOp::Scale().

6.7.2.142 void dsp_scale01f (float ∗, long)

clDSPOp::Scale01

Definition at line 948 of file cdspop.cc.

References clDSPOp::Scale01().

6.7.2.143 void dsp_scale01 (double ∗, long)

Definition at line 954 of file cdspop.cc.

References clDSPOp::Scale01().

6.7.2.144 void dsp_scale01f_nip (float ∗, const float ∗, long)

Definition at line 960 of file cdspop.cc.

References clDSPOp::Scale01().

6.7.2.145 void dsp_Scale01_nip (double ∗, const double ∗, long)

6.7.2.146 void dsp_sortf (float ∗, long)

clDSPOp::Sort

Definition at line 972 of file cdspop.cc.

References clDSPOp::Sort().

6.7.2.147 void dsp_sort (double ∗, long)

Definition at line 978 of file cdspop.cc.

References clDSPOp::Sort().

6.7.2.148 void dsp_sortl (long *, long)

Definition at line 984 of file cdspop.cc.

References clDSPOp::Sort().

6.7.2.149 void dsp_stddevf (float *, float *, const float *, long)

clDSPOp::StdDev

Definition at line 990 of file cdspop.cc.

References clDSPOp::StdDev().

6.7.2.150 void dsp_stddev (double *, double *, const double *, long)

Definition at line 997 of file cdspop.cc.

References clDSPOp::StdDev().

6.7.2.151 float dsp_sumf (const float *, long)

clDSPOp::Sum

Definition at line 1004 of file cdspop.cc.

References clDSPOp::Sum().

6.7.2.152 double dsp_sum (const double *, long)

Definition at line 1010 of file cdspop.cc.

References clDSPOp::Sum().

6.7.2.153 void dsp_squaref (float *, long)

clDSPOp::Square

Definition at line 1016 of file cdspop.cc.

References clDSPOp::Square().

6.7.2.154 void dsp_square (double *, long)

Definition at line 1022 of file cdspop.cc.

References clDSPOp::Square().

6.7.2.155 void dsp_squaref_nip (float ∗, const float ∗, long)

Definition at line 1028 of file cdspop.cc.

References clDSPOp::Square().

6.7.2.156 void dsp_square_nip (double ∗, const double ∗, long)

Definition at line 1034 of file cdspop.cc.

References clDSPOp::Square().

6.7.2.157 void dsp_convertu8f (float ∗, const unsigned char ∗, long)

clDSPOp::Convert

Definition at line 1040 of file cdspop.cc.

References clDSPOp::Convert().

6.7.2.158 void dsp_convertu8 (double ∗, const unsigned char ∗, long)

Definition at line 1046 of file cdspop.cc.

References clDSPOp::Convert().

6.7.2.159 void dsp_converts16f (float ∗, const signed short ∗, long, int)

Definition at line 1052 of file cdspop.cc.

References clDSPOp::Convert().

6.7.2.160 void dsp_converts16 (double ∗, const signed short ∗, long, int)

Definition at line 1059 of file cdspop.cc.

References clDSPOp::Convert().

6.7.2.161 void dsp_converts32f (float ∗, const signed int ∗, long, int)

Definition at line 1066 of file cdspop.cc.

References clDSPOp::Convert().

6.7.2.162 void dsp_converts32 (double ∗, const signed int ∗, long, int)

Definition at line 1073 of file cdspop.cc.

References clDSPOp::Convert().

6.7.2.163 void dsp_convertd64f (float ∗, const double ∗, long)

Definition at line 1080 of file cdspop.cc.

References clDSPOp::Convert().

6.7.2.164 void dsp_convertf32 (double ∗, const float ∗, long)

Definition at line 1086 of file cdspop.cc.

References clDSPOp::Convert().

6.7.2.165 void dsp_convertf32c (unsigned char ∗, const float ∗, long)

Definition at line 1092 of file cdspop.cc.

References clDSPOp::Convert().

6.7.2.166 void dsp_convertd64c (unsigned char ∗, const double ∗, long)

Definition at line 1098 of file cdspop.cc.

References clDSPOp::Convert().

6.7.2.167 void dsp_convertf32s (signed short ∗, const float ∗, long, int)

Definition at line 1104 of file cdspop.cc.

References clDSPOp::Convert().

6.7.2.168 void dsp_convertd64s (signed short ∗, const double ∗, long, int)

Definition at line 1111 of file cdspop.cc.

References clDSPOp::Convert().

6.7.2.169 void dsp_convertf32i (signed int ∗, const float ∗, long, int)

Definition at line 1118 of file cdspop.cc.

References clDSPOp::Convert().

6.7.2.170 void dsp_convertd64i (signed int ∗, const double ∗, long, int)

Definition at line 1125 of file cdspop.cc.

References clDSPOp::Convert().

6.7.2.171 void dsp_cart2polarf (float ∗, float ∗, const float ∗, const float ∗, long)

clDSPOp::CartToPolar

Definition at line 1132 of file cdspop.cc.

References clDSPOp::CartToPolar().

6.7.2.172 void dsp_cart2polar (double ∗, double ∗, const double ∗, const double ∗, long)

Definition at line 1139 of file cdspop.cc.

References clDSPOp::CartToPolar().

6.7.2.173 void dsp_cart2polar2f (float ∗, float ∗, const *stpSCplx*, long)

Definition at line 1146 of file cdspop.cc.

References clDSPOp::CartToPolar(), and stpSCplx.

6.7.2.174 void dsp_cart2polar2 (double ∗, double ∗, const *stpDCplx*, long)

Definition at line 1153 of file cdspop.cc.

References clDSPOp::CartToPolar(), and stpDCplx.

6.7.2.175 void dsp_cart2polar3f (stpSPolar, const *stpSCplx*, long)

Definition at line 1160 of file cdspop.cc.

References clDSPOp::CartToPolar(), stpSCplx, and stpSPolar.

6.7.2.176 void dsp_cart2polar3 (stpDPolar, const *stpDCplx*, long)

Definition at line 1166 of file cdspop.cc.

References clDSPOp::CartToPolar(), stpDCplx, and stpDPolar.

6.7.2.177 void dsp_cart2polar4f (utpSCoord, long)

Definition at line 1172 of file cdspop.cc.

References clDSPOp::CartToPolar(), and utpSCoord.

6.7.2.178 void dsp_cart2polar4 (utpDCoord, long)

Definition at line 1178 of file cdspop.cc.

References clDSPOp::CartToPolar(), and utpDCoord.

6.7.2.179 void dsp_polar2cartf (float *, float *, const float *, const float *, long)

clDSPOp::PolarToCart

Definition at line 1184 of file cdspop.cc.

References clDSPOp::PolarToCart().

6.7.2.180 void dsp_polar2cart (double *, double *, const double *, const double *, long)

Definition at line 1191 of file cdspop.cc.

References clDSPOp::PolarToCart().

6.7.2.181 void dsp_polar2cart2f (stpSCplx, const float *, const float *, long)

Definition at line 1198 of file cdspop.cc.

References clDSPOp::PolarToCart(), and stpSCplx.

6.7.2.182 void dsp_polar2cart2 (stpDCplx, const double *, const double *, long)

Definition at line 1205 of file cdspop.cc.

References clDSPOp::PolarToCart(), and stpDCplx.

6.7.2.183 void dsp_polar2cart3f (stpSCplx, const *stpSPolar*, long)

Definition at line 1212 of file cdspop.cc.

References clDSPOp::PolarToCart(), stpSCplx, and stpSPolar.

6.7.2.184 void dsp_polar2cart3 (stpDCplx, const *stpDPolar*, long)

Definition at line 1218 of file cdspop.cc.

References clDSPOp::PolarToCart(), stpDCplx, and stpDPolar.

6.7.2.185 void dsp_polar2cart4f (utpSCoord, long)

Definition at line 1224 of file cdspop.cc.

References clDSPOp::PolarToCart(), and utpSCoord.

6.7.2.186 void dsp_polar2cart4 (utpDCoord, long)

Definition at line 1230 of file cdspop.cc.

References clDSPOp::PolarToCart(), and utpDCoord.

6.7.2.187 float dsp_crosscorrf (const float $*$, const float $*$, long)

clDSPOp::CrossCorr

Definition at line 1236 of file cdspop.cc.

References clDSPOp::CrossCorr().

6.7.2.188 double dsp_crosscorr (const double $*$, const double $*$, long)

Definition at line 1242 of file cdspop.cc.

References clDSPOp::CrossCorr().

6.7.2.189 float dsp_crosscorr2f (const float $*$, const float $*$, long, long)

clDSPOp::DelCrossCorr

Definition at line 1248 of file cdspop.cc.

References clDSPOp::DelCrossCorr().

6.7.2.190 double dsp_crosscorr2 (const double $*$, const double $*$, long, long)

Definition at line 1255 of file cdspop.cc.

References clDSPOp::DelCrossCorr().

6.7.2.191 void dsp_crosscorr3f (float $*$, const float $*$, const float $*$, long, const long $*$, long)

Definition at line 1262 of file cdspop.cc.

References clDSPOp::DelCrossCorr().

6.7.2.192 void dsp_crosscorr3 (double $*$, const double $*$, const double $*$, long, const long $*$, long)

Definition at line 1270 of file cdspop.cc.

References clDSPOp::DelCrossCorr().

6.7.2.193 float dsp_energyf (const float $*$, long)

clDSPOp::Energy

Definition at line 1278 of file cdspop.cc.

References clDSPOp::Energy().

6.7.2.194 double dsp_energy (const double $*$, long)

Definition at line 1284 of file cdspop.cc.

References clDSPOp::Energy().

6.7.2.195 void dsp_magnitudef (float $*$, const *stpSCplx*, long)

clDSPOp::Magnitude

Definition at line 1290 of file cdspop.cc.

References clDSPOp::Magnitude(), and stpSCplx.

6.7.2.196 void dsp_magnitude (double $*$, const *stpDCplx*, long)

Definition at line 1296 of file cdspop.cc.

References clDSPOp::Magnitude(), and stpDCplx.

6.7.2.197 void dsp_powerf (float $*$, const *stpSCplx*, long)

clDSPOp::Power

Definition at line 1302 of file cdspop.cc.

References clDSPOp::Power(), and stpSCplx.

6.7.2.198 void dsp_power (double $*$, const *stpDCplx*, long)

Definition at line 1308 of file cdspop.cc.

References clDSPOp::Power(), and stpDCplx.

6.7.2.199 void dsp_phasef (float $*$, const *stpSCplx*, long)

clDSPOp::Phase

Definition at line 1314 of file cdspop.cc.

References clDSPOp::Phase(), and stpSCplx.

6.7.2.200 void dsp_phase (double $*$, const *stpDCplx*, long)

Definition at line 1320 of file cdspop.cc.

References clDSPOp::Phase(), and stpDCplx.

6.7.2.201 void dsp_powerphasef (float ∗, float ∗, const *stpSCplx*, long)

clDSPOp::PowerPhase

Definition at line 1326 of file cdspop.cc.

References clDSPOp::PowerPhase(), and stpSCplx.

6.7.2.202 void dsp_powerphase (double ∗, double ∗, const *stpDCplx*, long)

Definition at line 1333 of file cdspop.cc.

References clDSPOp::PowerPhase(), and stpDCplx.

6.7.2.203 void dsp_decimatef (float ∗, const float ∗, long, long)

clDSPOp::Decimate

Definition at line 1340 of file cdspop.cc.

References clDSPOp::Decimate().

6.7.2.204 void dsp_decimate (double ∗, const double ∗, long, long)

Definition at line 1347 of file cdspop.cc.

References clDSPOp::Decimate().

6.7.2.205 void dsp_decimateavgf (float ∗, const float ∗, long, long)

clDSPOp::DecimateAvg

Definition at line 1354 of file cdspop.cc.

References clDSPOp::DecimateAvg().

6.7.2.206 void dsp_decimateavg (double ∗, const double ∗, long, long)

Definition at line 1361 of file cdspop.cc.

References clDSPOp::DecimateAvg().

6.7.2.207 void dsp_interpolatef (float ∗, const float ∗, long, long)

clDSPOp::Interpolate

Definition at line 1368 of file cdspop.cc.

References clDSPOp::Interpolate().

6.7.2.208 void dsp_interpolate (double *, const double *, long, long)

Definition at line 1375 of file cdspop.cc.

References clDSPOp::Interpolate().

6.7.2.209 void dsp_interpolateavgf (float *, const float *, long, long)

clDSPOp::InterpolateAvg

Definition at line 1410 of file cdspop.cc.

References clDSPOp::InterpolateAvg().

6.7.2.210 void dsp_interpolateavg (double *, const double *, long, long)

Definition at line 1417 of file cdspop.cc.

References clDSPOp::InterpolateAvg().

6.7.2.211 void dsp_resamplef (float *, long, const float *, long)

clDSPOp::Resample

Definition at line 1382 of file cdspop.cc.

References clDSPOp::Resample().

6.7.2.212 void dsp_resample (double *, long, const double *, long)

Definition at line 1389 of file cdspop.cc.

References clDSPOp::Resample().

6.7.2.213 void dsp_resampleavgf (float *, long, const float *, long)

clDSPOp::ResampleAvg

Definition at line 1396 of file cdspop.cc.

References clDSPOp::ResampleAvg().

6.7.2.214 void dsp_resampleavg (double *, long, const double *, long)

Definition at line 1403 of file cdspop.cc.

References clDSPOp::ResampleAvg().

6.7.2.215 float dsp_rmsf (const float ∗, long)

clDSPOp::RMS

Definition at line 1424 of file cdspop.cc.

References clDSPOp::RMS().

6.7.2.216 double dsp_rms (const double ∗, long)

Definition at line 1430 of file cdspop.cc.

References clDSPOp::RMS().

6.7.2.217 float dsp_variancef (float ∗, float ∗, const float ∗, long)

clDSPOp::Variance

Definition at line 1436 of file cdspop.cc.

References clDSPOp::Variance().

6.7.2.218 double dsp_variance (double ∗, double ∗, const double ∗, long)

Definition at line 1443 of file cdspop.cc.

References clDSPOp::Variance().

6.7.2.219 float dsp_peaklevelf (const float ∗ *fpSrc*, long)

clDSPOp::PeakLevel

Definition at line 1450 of file cdspop.cc.

References clDSPOp::PeakLevel().

6.7.2.220 double dsp_peaklevel (const double ∗ *dpSrc*, long)

Definition at line 1456 of file cdspop.cc.

References clDSPOp::PeakLevel().

6.7.2.221 void dsp_mixf (float ∗, const float ∗, long)

clDSPOp::Mix

Definition at line 1462 of file cdspop.cc.

References clDSPOp::Mix().

6.7.2.222 void dsp_mix (double *, const double *, long)

Definition at line 1468 of file cdspop.cc.

References clDSPOp::Mix().

6.7.2.223 void dsp_mix2f (float *, const float *, const float *, long)

Definition at line 1474 of file cdspop.cc.

References clDSPOp::Mix().

6.7.2.224 void dsp_mix2 (double *, const double *, const double *, long)

Definition at line 1481 of file cdspop.cc.

References clDSPOp::Mix().

6.7.2.225 void dsp_mixnf (float *, const float *, long, long)

Definition at line 1488 of file cdspop.cc.

References clDSPOp::Mix().

6.7.2.226 void dsp_mixn (double *, const double *, long, long)

Definition at line 1495 of file cdspop.cc.

References clDSPOp::Mix().

6.7.2.227 void dsp_extractf (float *, const float *, long, long, long)

clDSPOp::Extract

Definition at line 1502 of file cdspop.cc.

References clDSPOp::Extract().

6.7.2.228 void dsp_extract (double *, const double *, long, long, long)

Definition at line 1509 of file cdspop.cc.

References clDSPOp::Extract().

6.7.2.229 void dsp_packf (float *, const float *, long, long, long)

clDSPOp::Pack

Definition at line 1516 of file cdspop.cc.

References clDSPOp::Pack().

6.7.2.230 void dsp_pack (double $*$, const double $*$, long, long, long)

Definition at line 1523 of file cdspop.cc.

References clDSPOp::Pack().

6.7.2.231 void dsp_fftw_convertf2cf (stpSCplx, const float $*$, long)

clDSPOp::FFTWConvert

Definition at line 1530 of file cdspop.cc.

References clDSPOp::FFTWConvert(), and stpSCplx.

6.7.2.232 void dsp_fftw_convertf2cd (stpDCplx, const float $*$, long)

Definition at line 1536 of file cdspop.cc.

References clDSPOp::FFTWConvert(), and stpDCplx.

6.7.2.233 void dsp_fftw_convertd2cf (stpSCplx, const double $*$, long)

Definition at line 1542 of file cdspop.cc.

References clDSPOp::FFTWConvert(), and stpSCplx.

6.7.2.234 void dsp_fftw_convertd2cd (stpDCplx, const double $*$, long)

Definition at line 1548 of file cdspop.cc.

References clDSPOp::FFTWConvert(), and stpDCplx.

6.7.2.235 void dsp_fftw_convertcf2f (float $*$, const *stpSCplx*, long)

Definition at line 1554 of file cdspop.cc.

References clDSPOp::FFTWConvert(), and stpSCplx.

6.7.2.236 void dsp_fftw_convertcd2f (float $*$, const *stpDCplx*, long)

Definition at line 1560 of file cdspop.cc.

References clDSPOp::FFTWConvert(), and stpDCplx.

6.7.2.237 void dsp_fftw_convertcf2d (double ∗, const *stpSCplx*, long)

Definition at line 1566 of file cdspop.cc.

References clDSPOp::FFTWConvert(), and stpSCplx.

6.7.2.238 void dsp_fftw_convertcd2d (double ∗, const *stpDCplx*, long)

Definition at line 1572 of file cdspop.cc.

References clDSPOp::FFTWConvert(), and stpDCplx.

6.7.2.239 dsp_t dsp_new (void)

Creates new instance of DSP object.

Returns:
DSP object instance

Definition at line 1578 of file cdspop.cc.

References dsp_t.

6.7.2.240 void dsp_delete (dsp_t)

Deletes DSP object instance.

Parameters:
dspInst DSP object instance

Definition at line 1584 of file cdspop.cc.

References dsp_t.

6.7.2.241 void dsp_win_bartlettf (dsp_t, float ∗, long)

clDSPOp::WinBartlett

Definition at line 1590 of file cdspop.cc.

References dsp_t.

6.7.2.242 void dsp_win_bartlett (dsp_t, double ∗, long)

Definition at line 1596 of file cdspop.cc.

References dsp_t.

6.7.2.243 void dsp_win_blackmanf (dsp_t, float *, long, float)

clDSPOp::WinBlackman

Definition at line 1602 of file cdspop.cc.

References dsp_t.

6.7.2.244 void dsp_win_blackman (dsp_t, double *, long, double)

Definition at line 1609 of file cdspop.cc.

References dsp_t.

6.7.2.245 void dsp_win_blackman_harrisf (dsp_t, float *, long)

clDSPOp::WinBlackmanHarris

Definition at line 1616 of file cdspop.cc.

References dsp_t.

6.7.2.246 void dsp_win_blackman_harris (dsp_t, double *, long)

Definition at line 1622 of file cdspop.cc.

References dsp_t.

6.7.2.247 void dsp_win_cos_taperedf (dsp_t, float *, long)

clDSPOp::WinCosTapered

Definition at line 1628 of file cdspop.cc.

References dsp_t.

6.7.2.248 void dsp_win_cos_tapered (dsp_t, double *, long)

Definition at line 1634 of file cdspop.cc.

References dsp_t.

6.7.2.249 void dsp_win_exact_blackmanf (dsp_t, float *, long)

clDSPOp::WinExactBlackman

Definition at line 1640 of file cdspop.cc.

References dsp_t.

6.7.2.250 void dsp_win_exact_blackman (dsp_t, double *, long)

Definition at line 1646 of file cdspop.cc.

References dsp_t.

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

Definition at line 1652 of file cdspop.cc.

References dsp_t.

6.7.2.254 void dsp_win_flat_top (dsp_t, double *, long)

Definition at line 1658 of file cdspop.cc.

References dsp_t.

6.7.2.255 void dsp_win_generic_cosf (dsp_t, float *, long, const float *, long)

clDSPOp::WinGenericCos

Definition at line 1664 of file cdspop.cc.

References dsp_t.

6.7.2.256 void dsp_win_generic_cos (dsp_t, double *, long, const double *, long)

Definition at line 1672 of file cdspop.cc.

References dsp_t.

6.7.2.257 void dsp_win_hammingf (dsp_t, float *, long)

clDSPOp::WinHamming

Definition at line 1680 of file cdspop.cc.

References dsp_t.

6.7.2.258 void dsp_win_hamming (dsp_t, double *, long)

Definition at line 1686 of file cdspop.cc.

References dsp_t.

6.7.2.259 void dsp_win_hanningf (dsp_t, float *, long)

clDSPOp::WinHanning

Definition at line 1692 of file cdspop.cc.

References dsp_t.

6.7.2.260 void dsp_win_hanning (dsp_t, double *, long)

Definition at line 1698 of file cdspop.cc.

References dsp_t.

6.7.2.261 void dsp_win_kaiserf (dsp_t, float *, float, long)

clDSPOp::WinKaiser

Definition at line 1704 of file cdspop.cc.

References dsp_t.

6.7.2.262 void dsp_win_kaiser (dsp_t, double *, double, long)

Definition at line 1710 of file cdspop.cc.

References dsp_t.

6.7.2.263 void dsp_win_kaiser_besself (dsp_t, float *, float, long)

clDSPOp::WinKaiserBessel

Definition at line 1716 of file cdspop.cc.

References dsp_t.

6.7.2.264 void dsp_win_kaiser_bessel (dsp_t, double *, double, long)

Definition at line 1723 of file cdspop.cc.

References dsp_t.

6.7.2.265 void dsp_win_tukeyf (dsp_t, float *, long)

clDSPOp::WinTukey

Definition at line 1730 of file cdspop.cc.

References dsp_t.

6.7.2.266 void dsp_win_tukey (dsp_t, double *, long)

Definition at line 1736 of file cdspop.cc.

References dsp_t.

6.7.2.267 void dsp_win_dolph_chebyshevf (dsp_t, float *, float, long)

clDSPOp::WinDolphChebyshev

Definition at line 1742 of file cdspop.cc.

References dsp_t.

6.7.2.268 void dsp_win_dolph_chebyshev (dsp_t, double *, double, long)

Definition at line 1749 of file cdspop.cc.

References dsp_t.

6.7.2.269 long dsp_rebufferf (dsp_t, float *, const float *, long, long)

clDSPOp::ReBuffer

Definition at line 1756 of file cdspop.cc.

References dsp_t.

6.7.2.270 long dsp_rebuffer (dsp_t, double *, const double *, long, long)

Definition at line 1764 of file cdspop.cc.

References dsp_t.

6.7.2.271 float dsp_deg2radf (dsp_t, float)

clDSPOp::DegToRad

Definition at line 1772 of file cdspop.cc.

References dsp_t.

6.7.2.272 double dsp_deg2rad (dsp_t, double)

Definition at line 1778 of file cdspop.cc.

References dsp_t.

6.7.2.273 float dsp_rad2degf (dsp_t, float)

clDSPOp::RadToDeg

Definition at line 1784 of file cdspop.cc.

References dsp_t.

6.7.2.274 double dsp_rad2deg (dsp_t, double)

Definition at line 1790 of file cdspop.cc.

References dsp_t.

6.7.2.275 void dsp_fir_allocatef (dsp_t, const float $*$, long)

clDSPOp::FIRAllocate

Definition at line 1796 of file cdspop.cc.

References dsp_t.

6.7.2.276 void dsp_fir_allocate (dsp_t, const double $*$, long)

Definition at line 1802 of file cdspop.cc.

References dsp_t.

6.7.2.277 void dsp_fir_free (dsp_t)

clDSPOp::FIRFree

Definition at line 1808 of file cdspop.cc.

References dsp_t.

6.7.2.278 void dsp_fir_filterf (dsp_t, float $*$, long)

clDSPOp::FIRFilter

Definition at line 1814 of file cdspop.cc.

References dsp_t.

6.7.2.279 void dsp_fir_filter (dsp_t, double *, long)

Definition at line 1820 of file cdspop.cc.

References dsp_t.

6.7.2.280 void dsp_fir_filterf_nip (dsp_t, float *, const float *, long)

Definition at line 1826 of file cdspop.cc.

References dsp_t.

6.7.2.281 void dsp_fir_filter_nip (dsp_t, double *, const double *, long)

Definition at line 1833 of file cdspop.cc.

References dsp_t.

6.7.2.282 void dsp_fir_filterf_fst (dsp_t, float *, float *, long)

clDSPOp::FIRFilterF

Definition at line 1840 of file cdspop.cc.

References dsp_t.

6.7.2.283 void dsp_fir_filter_fst (dsp_t, double *, double *, long)

Definition at line 1847 of file cdspop.cc.

References dsp_t.

6.7.2.284 void dsp_iir_initf (dsp_t, const float *)

clDSPOp::IIRInitialize

Definition at line 1854 of file cdspop.cc.

References dsp_t.

6.7.2.285 void dsp_iir_init (dsp_t, const double *)

Definition at line 1860 of file cdspop.cc.

References dsp_t.

6.7.2.286 void dsp_iir_filterf (dsp_t, float *, long)

clDSPOp::IIRFilter

Definition at line 1866 of file cdspop.cc.

References dsp_t.

6.7.2.287 void dsp_iir_filter (dsp_t, double *, long)

Definition at line 1872 of file cdspop.cc.

References dsp_t.

6.7.2.288 void dsp_iir_filterf_nip (dsp_t, float *, const float *, long)

Definition at line 1878 of file cdspop.cc.

References dsp_t.

6.7.2.289 void dsp_iir_filter_nip (dsp_t, double *, const double *, long)

Definition at line 1885 of file cdspop.cc.

References dsp_t.

6.7.2.290 void dsp_iir_clear (dsp_t)

clDSPOp::IIRClear

Definition at line 1892 of file cdspop.cc.

References dsp_t.

6.7.2.291 void dsp_fft_init (dsp_t, long, int)

clDSPOp::FFTInitialize

Definition at line 1898 of file cdspop.cc.

References dsp_t.

6.7.2.292 void dsp_fft_uninit (dsp_t)

clDSPOp::FFTUninitialize

Definition at line 1904 of file cdspop.cc.

References dsp_t.

6.7.2.293 void dsp_fftf (dsp_t, stpSCplx, float *)

clDSPOp::FFTi

Definition at line 1910 of file cdspop.cc.

References dsp_t, and stpSCplx.

6.7.2.294 void dsp_fft (dsp_t, stpDCplx, double ∗)

Definition at line 1916 of file cdspop.cc.

References dsp_t, and stpDCplx.

6.7.2.295 void dsp_fftf_nip (dsp_t, stpSCplx, const float ∗)

clDSPOp::FFTo

Definition at line 1922 of file cdspop.cc.

References dsp_t, and stpSCplx.

6.7.2.296 void dsp_fft_nip (dsp_t, stpDCplx, const double ∗)

Definition at line 1928 of file cdspop.cc.

References dsp_t, and stpDCplx.

6.7.2.297 void dsp_cfftf_nip (dsp_t, stpSCplx, const *stpSCplx*)

Definition at line 1934 of file cdspop.cc.

References dsp_t, and stpSCplx.

6.7.2.298 void dsp_cfft_nip (dsp_t, stpDCplx, const *stpDCplx*)

Definition at line 1940 of file cdspop.cc.

References dsp_t, and stpDCplx.

6.7.2.299 void dsp_ifftf_nip (dsp_t, float ∗, const *stpSCplx*)

clDSPOp::IFFTo

Definition at line 1946 of file cdspop.cc.

References dsp_t, and stpSCplx.

6.7.2.300 void dsp_ifft_nip (dsp_t, double ∗, const *stpDCplx*)

Definition at line 1952 of file cdspop.cc.

References dsp_t, and stpDCplx.

6.7.2.301 void dsp_cifftf_nip (dsp_t, stpSCplx, const *stpSCplx*)

Definition at line 1958 of file cdspop.cc.

References dsp_t, and stpSCplx.

6.7.2.302 void dsp_cifft_nip (dsp_t, stpDCplx, const *stpDCplx*)

Definition at line 1964 of file cdspop.cc.

References dsp_t, and stpDCplx.

6.7.2.303 void dsp_dct_init (dsp_t, long)

clDSPOp::DCTInitialize

Definition at line 1970 of file cdspop.cc.

References dsp_t.

6.7.2.304 void dsp_dct_uninit (dsp_t)

clDSPOp::DCTUninitialize

Definition at line 1976 of file cdspop.cc.

References dsp_t.

6.7.2.305 void dsp_dctf (dsp_t, float ∗)

clDSPOp::DCTi

Definition at line 1982 of file cdspop.cc.

References dsp_t.

6.7.2.306 void dsp_dct (dsp_t, double ∗)

Definition at line 1988 of file cdspop.cc.

References dsp_t.

6.7.2.307 void dsp_dctf_nip (dsp_t, float ∗, const float ∗)

clDSPOp::DCTo

Definition at line 1994 of file cdspop.cc.

References dsp_t.

6.7.2.308 void dsp_dct_nip (dsp_t, double ∗, const double ∗)

Definition at line 2000 of file cdspop.cc.

References dsp_t.

6.7.2.309 void dsp_idctf (dsp_t, float ∗)

clDSPOp::IDCTi

Definition at line 2006 of file cdspop.cc.

References dsp_t.

6.7.2.310 void dsp_idct (dsp_t, double ∗)

Definition at line 2012 of file cdspop.cc.

References dsp_t.

6.7.2.311 void dsp_idctf_nip (dsp_t, float ∗, const float ∗)

clDSPOp::IDCTo

Definition at line 2018 of file cdspop.cc.

References dsp_t.

6.7.2.312 void dsp_idct_nip (dsp_t, double ∗, const double ∗)

Definition at line 2024 of file cdspop.cc.

References dsp_t.

6.7.2.313 dsp_iircas_t dsp_iir_cas_new (void)

Creates new instance of cascaded IIR object.

Returns:

Cascade IIR instance

Definition at line 2033 of file cdspop.cc.

References dsp_iircas_t.

6.7.2.314 void dsp_iir_cas_delete (dsp_iircas_t)

Deletes cascaded IIR object instance.

Parameters:
dspiircasInst Cascaded IIR instance

Definition at line 2039 of file cdspop.cc.

References dsp_iircas_t.

6.7.2.315 int dsp_iir_cas_initf (dsp_iircas_t, const *float*[ ][5], long)

clIIRCascade::Initialize

Definition at line 2045 of file cdspop.cc.

References dsp_iircas_t.

6.7.2.316 int dsp_iir_cas_init (dsp_iircas_t, const *double*[ ][5], long)

Definition at line 2054 of file cdspop.cc.

References dsp_iircas_t.

6.7.2.317 void dsp_iir_cas_uninit (dsp_iircas_t)

Definition at line 2063 of file cdspop.cc.

References dsp_iircas_t.

6.7.2.318 void dsp_iir_cas_processf (dsp_iircas_t, float *, long)

Definition at line 2069 of file cdspop.cc.

References dsp_iircas_t.

6.7.2.319 void dsp_iir_cas_process (dsp_iircas_t, double *, long)

Definition at line 2076 of file cdspop.cc.

References dsp_iircas_t.

6.7.2.320 void dsp_iir_cas_processf_nip (dsp_iircas_t, float *, const float *, long)

Definition at line 2083 of file cdspop.cc.

References dsp_iircas_t.

6.7.2.321 void dsp_iir_cas_process_nip (dsp_iircas_t, double *, const double *, long)

Definition at line 2090 of file cdspop.cc.

References dsp_iircas_t.

6.7.2.322 void dsp_iir_cas_clear (dsp_iircas_t)

Definition at line 2097 of file cdspop.cc.

References dsp_iircas_t.

6.7.2.323 dsp_decfft_t dsp_dec_fft_new (void)

Creates new instance of FFT decimator object.

Returns:

FFT decimator instance

Definition at line 2106 of file cdspop.cc.

References dsp_decfft_t.

6.7.2.324 void dsp_dec_fft_delete (dsp_decfft_t)

Deletes FFT decimator object instance.

Parameters:

dspdecfftInst FFT decimator instance

Definition at line 2112 of file cdspop.cc.

References dsp_decfft_t.

6.7.2.325 int dsp_dec_fft_initf (dsp_decfft_t, long, long, int)

clFFTDecimator::Initialize

Definition at line 2118 of file cdspop.cc.

References dsp_decfft_t.

6.7.2.326 int dsp_dec_fft_init (dsp_decfft_t, long, long, int)

Definition at line 2129 of file cdspop.cc.

References dsp_decfft_t.

6.7.2.327 void dsp_dec_fft_uninit (dsp_decfft_t)

clFFTDecimator::Uninitialize

Definition at line 2140 of file cdspop.cc.

References dsp_decfft_t.

6.7.2.328 void dsp_dec_fft_putf (dsp_decfft_t, const float *, long)

clFFTDecimator::Put

Definition at line 2146 of file cdspop.cc.

References dsp_decfft_t.

6.7.2.329 void dsp_dec_fft_put (dsp_decfft_t, const double *, long)

Definition at line 2153 of file cdspop.cc.

References dsp_decfft_t.

6.7.2.330 int dsp_dec_fft_getf (dsp_decfft_t, float *, long)

clFFTDecimator::Get

Definition at line 2160 of file cdspop.cc.

References dsp_decfft_t.

6.7.2.331 int dsp_dec_fft_get (dsp_decfft_t, double *, long)

Definition at line 2169 of file cdspop.cc.

References dsp_decfft_t.

6.7.2.332 dsp_decfir_t dsp_dec_fir_new (void)

Creates new instance of FIR decimator object.

Returns:
FIR decimator instance

Definition at line 2181 of file cdspop.cc.

References dsp_decfir_t.

6.7.2.333 void dsp_dec_fir_delete (dsp_decfir_t)

Deletes FIR decimator object instance.

Parameters:

dspdecfirInst FIR decimator instance

Definition at line 2187 of file cdspop.cc.

References dsp_decfir_t.

6.7.2.334 int dsp_dec_fir_initf (dsp_decfir_t, long, int)

clFIRDecimator::Initialize

Definition at line 2193 of file cdspop.cc.

References dsp_decfir_t.

6.7.2.335 int dsp_dec_fir_init (dsp_decfir_t, long, int)

Definition at line 2203 of file cdspop.cc.

References dsp_decfir_t.

6.7.2.336 void dsp_dec_fir_uninit (dsp_decfir_t)

clFIRDecimator::Uninitialize

Definition at line 2213 of file cdspop.cc.

References dsp_decfir_t.

6.7.2.337 void dsp_dec_fir_putf (dsp_decfir_t, const float ∗, long)

clFIRDecimator::Put

Definition at line 2219 of file cdspop.cc.

References dsp_decfir_t.

6.7.2.338 void dsp_dec_fir_put (dsp_decfir_t, const double ∗, long)

Definition at line 2226 of file cdspop.cc.

References dsp_decfir_t.

6.7.2.339 int dsp_dec_fir_getf (dsp_decfir_t, float ∗, long)

clFIRDecimator::Get

Definition at line 2233 of file cdspop.cc.

References dsp_decfir_t.

6.7.2.340 int dsp_dec_fir_get (dsp_decfir_t, double *, long)

Definition at line 2242 of file cdspop.cc.

References dsp_decfir_t.

6.7.2.341 dsp_deciir_t dsp_dec_iir_new (void)

Creates new instance of IIR decimator object.

Returns:
IIR decimator instance

Definition at line 2254 of file cdspop.cc.

References dsp_deciir_t.

6.7.2.342 void dsp_dec_iir_delete (dsp_deciir_t)

Deletes IIR decimator object instance.

Parameters:
dspdeciirInst IIR decimator instance

Definition at line 2260 of file cdspop.cc.

References dsp_deciir_t.

6.7.2.343 int dsp_dec_iir_initf (dsp_deciir_t, long, int)

clIIRDecimator::Initialize

Definition at line 2266 of file cdspop.cc.

References dsp_deciir_t.

6.7.2.344 int dsp_dec_iir_init (dsp_deciir_t, long, int)

Definition at line 2276 of file cdspop.cc.

References dsp_deciir_t.

6.7.2.345 void dsp_dec_iir_uninit (dsp_deciir_t)

clIIRDecimator::Uninitialize

Definition at line 2286 of file cdspop.cc.

References dsp_deciir_t.

6.7.2.346 void dsp_dec_iir_putf (dsp_deciir_t, const float *, long)

clIIRDecimator::Put

Definition at line 2292 of file cdspop.cc.

References dsp_deciir_t.

6.7.2.347 void dsp_dec_iir_put (dsp_deciir_t, const double *, long)

Definition at line 2299 of file cdspop.cc.

References dsp_deciir_t.

6.7.2.348 int dsp_dec_iir_getf (dsp_deciir_t, float *, long)

clIIRDecimator::Get

Definition at line 2306 of file cdspop.cc.

References dsp_deciir_t.

6.7.2.349 int dsp_dec_iir_get (dsp_deciir_t, double *, long)

Definition at line 2315 of file cdspop.cc.

References dsp_deciir_t.

6.7.2.350 dsp_decrec_t dsp_dec_rec_new (void)

Creates new instance of recursive decimator object.

Returns:
Recursive decimator instance

Definition at line 2327 of file cdspop.cc.

References dsp_decrec_t.

6.7.2.351 void dsp_dec_rec_delete (dsp_decrec_t)

Deletes recursive decimator object instance.

Parameters:
dspdecrecInst Recursive decimator instance

Definition at line 2333 of file cdspop.cc.

References dsp_decrec_t.

6.7.2.352 int dsp_dec_rec_initf (dsp_decrec_t, long, long, float, int)

clRecDecimator::Initialize

Definition at line 2339 of file cdspop.cc.

References dsp_decrec_t.

6.7.2.353 int dsp_dec_rec_init (dsp_decrec_t, long, long, double, int)

Definition at line 2351 of file cdspop.cc.

References dsp_decrec_t.

6.7.2.354 void dsp_dec_rec_uninit (dsp_decrec_t)

clRecDecimator::Uninitialize

Definition at line 2363 of file cdspop.cc.

References dsp_decrec_t.

6.7.2.355 void dsp_dec_rec_putf (dsp_decrec_t, const float *, long)

clRecDecimator::Put

Definition at line 2369 of file cdspop.cc.

References dsp_decrec_t.

6.7.2.356 void dsp_dec_rec_put (dsp_decrec_t, const double *, long)

Definition at line 2376 of file cdspop.cc.

References dsp_decrec_t.

6.7.2.357 int dsp_dec_rec_getf (dsp_decrec_t, float *, long)

clRecDecimator::Get

Definition at line 2383 of file cdspop.cc.

References dsp_decrec_t.

6.7.2.358 int dsp_dec_rec_get (dsp_decrec_t, double *, long)

Definition at line 2392 of file cdspop.cc.

References dsp_decrec_t.

6.7.2.359 dsp_intfft_t dsp_int_fft_new (void)

Creates new instance of FFT interpolator object.

Returns:

FFT interpolator instance

Definition at line 2404 of file cdspop.cc.

References dsp_intfft_t.

6.7.2.360 void dsp_int_fft_delete (dsp_intfft_t)

Deletes FFT interpolator object instance.

Parameters:

dspintfftInst FFT interpolator instance

Definition at line 2410 of file cdspop.cc.

References dsp_intfft_t.

6.7.2.361 int dsp_int_fft_initf (dsp_intfft_t, long, long, int)

clFFTInterpolator::Initialize

Definition at line 2416 of file cdspop.cc.

References dsp_intfft_t.

6.7.2.362 int dsp_int_fft_init (dsp_intfft_t, long, long, int)

Definition at line 2427 of file cdspop.cc.

References dsp_intfft_t.

6.7.2.363 void dsp_int_fft_uninit (dsp_intfft_t)

clFFTInterpolator::Uninitialize

Definition at line 2438 of file cdspop.cc.

References dsp_intfft_t.

6.7.2.364 void dsp_int_fft_putf (dsp_intfft_t, const float *, long)

clFFTInterpolator::Put

Definition at line 2444 of file cdspop.cc.

References dsp_intfft_t.

6.7.2.365 void dsp_int_fft_put (dsp_intfft_t, const double *, long)

Definition at line 2451 of file cdspop.cc.

References dsp_intfft_t.

6.7.2.366 int dsp_int_fft_getf (dsp_intfft_t, float *, long)

clFFTInterpolator::Get

Definition at line 2458 of file cdspop.cc.

References dsp_intfft_t.

6.7.2.367 int dsp_int_fft_get (dsp_intfft_t, double *, long)

Definition at line 2467 of file cdspop.cc.

References dsp_intfft_t.

6.7.2.368 dsp_intfir_t dsp_int_fir_new (void)

Creates new instance of FIR interpolator object.

Returns:

FIR interpolator instance

Definition at line 2479 of file cdspop.cc.

References dsp_intfir_t.

6.7.2.369 void dsp_int_fir_delete (dsp_intfir_t)

Deletes FIR interpolator object instance.

Parameters:

dspintfirInst FIR interpolator instance

Definition at line 2485 of file cdspop.cc.

References dsp_intfir_t.

6.7.2.370 int dsp_int_fir_initf (dsp_intfir_t, long, int)

clFIRInterpolator::Initialize

Definition at line 2491 of file cdspop.cc.

References dsp_intfir_t.

6.7.2.371 int dsp_int_fir_init (dsp_intfir_t, long, int)

Definition at line 2501 of file cdspop.cc.

References dsp_intfir_t.

6.7.2.372 void dsp_int_fir_uninit (dsp_intfir_t)

clFIRInterpolator::Uninitialize

Definition at line 2511 of file cdspop.cc.

References dsp_intfir_t.

6.7.2.373 void dsp_int_fir_putf (dsp_intfir_t, const float *, long)

clFIRInterpolator::Put

Definition at line 2517 of file cdspop.cc.

References dsp_intfir_t.

6.7.2.374 void dsp_int_fir_put (dsp_intfir_t, const double *, long)

Definition at line 2524 of file cdspop.cc.

References dsp_intfir_t.

6.7.2.375 int dsp_int_fir_getf (dsp_intfir_t, float *, long)

clFIRInterpolator::Get

Definition at line 2531 of file cdspop.cc.

References dsp_intfir_t.

6.7.2.376 int dsp_int_fir_get (dsp_intfir_t, double *, long)

Definition at line 2540 of file cdspop.cc.

References dsp_intfir_t.

6.7.2.377 dsp_intiir_t dsp_int_iir_new (void)

Creates new instance of IIR interpolator object.

Returns:

IIR interpolator instance

Definition at line 2552 of file cdspop.cc.

References dsp_intiir_t.

6.7.2.378 void dsp_int_iir_delete (dsp_intiir_t)

Deletes IIR interpolator object instance.

Parameters:

dspintiirInst IIR interpolator instance

Definition at line 2558 of file cdspop.cc.

References dsp_intiir_t.

6.7.2.379 int dsp_int_iir_initf (dsp_intiir_t, long, int)

clIIRInterpolator::Initialize

Definition at line 2564 of file cdspop.cc.

References dsp_intiir_t.

6.7.2.380 int dsp_int_iir_init (dsp_intiir_t, long, int)

Definition at line 2574 of file cdspop.cc.

References dsp_intiir_t.

6.7.2.381 void dsp_int_iir_uninit (dsp_intiir_t)

clIIRInterpolator::Uninitialize

Definition at line 2584 of file cdspop.cc.

References dsp_intiir_t.

6.7.2.382 void dsp_int_iir_putf (dsp_intiir_t, const float *, long)

clIIRInterpolator::Put

Definition at line 2590 of file cdspop.cc.

References dsp_intiir_t.

6.7.2.383 void dsp_int_iir_put (dsp_intiir_t, const double *, long)

Definition at line 2597 of file cdspop.cc.

References dsp_intiir_t.

6.7.2.384 int dsp_int_iir_getf (dsp_intiir_t, float *, long)

clIIRInterpolator::Get

Definition at line 2604 of file cdspop.cc.

References dsp_intiir_t.

6.7.2.385 int dsp_int_iir_get (dsp_intiir_t, double *, long)

Definition at line 2613 of file cdspop.cc.

References dsp_intiir_t.

6.7.2.386 dsp_intrec_t dsp_int_rec_new (void)

Creates new instance of recursive interpolator object.

Returns:
Recursive interpolator instance

Definition at line 2625 of file cdspop.cc.

References dsp_intrec_t.

6.7.2.387 void dsp_int_rec_delete (dsp_intrec_t)

Deletes recursive interpolator object instance.

Parameters:
dspdecrecInst Recursive interpolator instance

Definition at line 2631 of file cdspop.cc.

References dsp_intrec_t.

6.7.2.388 int dsp_int_rec_initf (dsp_intrec_t, long, long, float, int)

clRecInterpolator::Initialize

Definition at line 2637 of file cdspop.cc.

References dsp_intrec_t.

6.7.2.389 int dsp_int_rec_init (dsp_intrec_t, long, long, double, int)

Definition at line 2649 of file cdspop.cc.

References dsp_intrec_t.

6.7.2.390 void dsp_int_rec_uninit (dsp_intrec_t)

clRecInterpolator::Uninitialize

Definition at line 2661 of file cdspop.cc.

References dsp_intrec_t.

6.7.2.391 void dsp_int_rec_putf (dsp_intrec_t, const float $*$, long)

clRecInterpolator::Put

Definition at line 2667 of file cdspop.cc.

References dsp_intrec_t.

6.7.2.392 void dsp_int_rec_put (dsp_intrec_t, const double $*$, long)

Definition at line 2674 of file cdspop.cc.

References dsp_intrec_t.

6.7.2.393 int dsp_int_rec_getf (dsp_intrec_t, float $*$, long)

clRecInterpolator::Get

Definition at line 2681 of file cdspop.cc.

References dsp_intrec_t.

6.7.2.394 int dsp_int_rec_get (dsp_intrec_t, double $*$, long)

Definition at line 2690 of file cdspop.cc.

References dsp_intrec_t.

6.7.2.395 dsp_filter_t dsp_filter_new (void)

Creates new instance of FFT filter object.

Returns:
FFT filter instance

Definition at line 2702 of file cdspop.cc.

References dsp_filter_t.

6.7.2.396 void dsp_filter_delete (dsp_filter_t)

Deletes instance of FFT filter object.

Parameters:

dspfilterInst FFT filter instance

Definition at line 2708 of file cdspop.cc.

References dsp_filter_t.

6.7.2.397 int dsp_filter_initf (dsp_filter_t, long)

clFilter::Initialize

Definition at line 2714 of file cdspop.cc.

References dsp_filter_t.

6.7.2.398 int dsp_filter_init (dsp_filter_t, long)

Definition at line 2724 of file cdspop.cc.

References dsp_filter_t.

6.7.2.399 int dsp_filter_init2f (dsp_filter_t, long, const float *, float, float)

Definition at line 2734 of file cdspop.cc.

References dsp_filter_t.

6.7.2.400 int dsp_filter_init2 (dsp_filter_t, long, const double *, double, double)

Definition at line 2743 of file cdspop.cc.

References dsp_filter_t.

6.7.2.401 int dsp_filter_init_lpf (dsp_filter_t, float, float, float, float)

clFilter::InitializeLP

Definition at line 2752 of file cdspop.cc.

References dsp_filter_t.

6.7.2.402 int dsp_filter_init_lp (dsp_filter_t, double, double, double, double)

Definition at line 2761 of file cdspop.cc.

References dsp_filter_t.

6.7.2.403 int dsp_filter_init_hpf (dsp_filter_t, float, float, float, float)

clFilter::InitializeHP

Definition at line 2770 of file cdspop.cc.

References dsp_filter_t.

6.7.2.404 int dsp_filter_init_hp (dsp_filter_t, double, double, double, double)

Definition at line 2779 of file cdspop.cc.

References dsp_filter_t.

6.7.2.405 void dsp_filter_set_coeffsf (dsp_filter_t, const float ∗)

clFilter::SetCoeffs

Definition at line 2788 of file cdspop.cc.

References dsp_filter_t.

6.7.2.406 void dsp_filter_set_ceoffs (dsp_filter_t, const double ∗)

6.7.2.407 void dsp_filter_set_ccoeffsf (dsp_filter_t, const *stpSCplx*, int)

Definition at line 2802 of file cdspop.cc.

References dsp_filter_t, and stpSCplx.

6.7.2.408 void dsp_filter_set_ccoeffs (dsp_filter_t, const *stpDCplx*, int)

Definition at line 2810 of file cdspop.cc.

References dsp_filter_t, and stpDCplx.

6.7.2.409 void dsp_filter_get_coeffsf (dsp_filter_t, float ∗)

clFilter::GetCoeffs

Definition at line 2818 of file cdspop.cc.

References dsp_filter_t.

6.7.2.410 void dsp_filter_get_coeffs (dsp_filter_t, double ∗)

Definition at line 2825 of file cdspop.cc.

References dsp_filter_t.

6.7.2.411 void dsp_filter_get_ccoeffsf (dsp_filter_t, stpSCplx)

Definition at line 2832 of file cdspop.cc.

References dsp_filter_t, and stpSCplx.

6.7.2.412 void dsp_filter_get_ccoeffs (dsp_filter_t, stpDCplx)

Definition at line 2839 of file cdspop.cc.

References dsp_filter_t, and stpDCplx.

6.7.2.413 void dsp_filter_putf (dsp_filter_t, const float ∗, long)

clFilter::Put

Definition at line 2846 of file cdspop.cc.

References dsp_filter_t.

6.7.2.414 void dsp_filter_put (dsp_filter_t, const double ∗, long)

Definition at line 2853 of file cdspop.cc.

References dsp_filter_t.

6.7.2.415 void dsp_filter_put2f (dsp_filter_t, const float ∗, long, const *stpSCplx*)

Definition at line 2860 of file cdspop.cc.

References dsp_filter_t, and stpSCplx.

6.7.2.416 void dsp_filter_put2 (dsp_filter_t, const double ∗, long, const *stpDCplx*)

Definition at line 2867 of file cdspop.cc.

References dsp_filter_t, and stpDCplx.

6.7.2.417 void dsp_filter_getf (dsp_filter_t, float ∗, long)

clFilter::Get

Definition at line 2874 of file cdspop.cc.

References dsp_filter_t.

6.7.2.418 void dsp_filter_get (dsp_filter_t, double ∗, long)

Definition at line 2881 of file cdspop.cc.

References dsp_filter_t.

6.7.2.419 void dsp_filter_design_lpf (dsp_filter_t, float *, int)

clFilter::DesignLP

Definition at line 2888 of file cdspop.cc.

References dsp_filter_t.

6.7.2.420 void dsp_filter_design_lp (dsp_filter_t, double *, int)

Definition at line 2896 of file cdspop.cc.

References dsp_filter_t.

6.7.2.421 void dsp_filter_design_lp2f (dsp_filter_t, float *, float, int)

Definition at line 2904 of file cdspop.cc.

References dsp_filter_t.

6.7.2.422 void dsp_filter_design_lp2 (dsp_filter_t, double *, double, int)

Definition at line 2912 of file cdspop.cc.

References dsp_filter_t.

6.7.2.423 void dsp_filter_design_hpf (dsp_filter_t, float *)

clFilter::DesignHP

Definition at line 2920 of file cdspop.cc.

References dsp_filter_t.

6.7.2.424 void dsp_filter_design_hp (dsp_filter_t, double *)

Definition at line 2926 of file cdspop.cc.

References dsp_filter_t.

6.7.2.425 void dsp_filter_design_hp2f (dsp_filter_t, float *, float)

Definition at line 2932 of file cdspop.cc.

References dsp_filter_t.

6.7.2.426 void dsp_filter_design_hp2 (dsp_filter_t, double *, double)

Definition at line 2939 of file cdspop.cc.

References dsp_filter_t.

6.7.2.427 void dsp_filter_design_bpf (dsp_filter_t, float *, float *)

clFilter::DesignBP

Definition at line 2946 of file cdspop.cc.

References dsp_filter_t.

6.7.2.428 void dsp_filter_design_bp (dsp_filter_t, double *, double *)

Definition at line 2953 of file cdspop.cc.

References dsp_filter_t.

6.7.2.429 void dsp_filter_design_bp2f (dsp_filter_t, float *, float *, float)

Definition at line 2960 of file cdspop.cc.

References dsp_filter_t.

6.7.2.430 void dsp_filter_design_bp2 (dsp_filter_t, double *, double *, double)

Definition at line 2968 of file cdspop.cc.

References dsp_filter_t.

6.7.2.431 void dsp_filter_design_brf (dsp_filter_t, float *, float *)

clFilter::DesignBR

Definition at line 2976 of file cdspop.cc.

References dsp_filter_t.

6.7.2.432 void dsp_filter_design_br (dsp_filter_t, double *, double *)

Definition at line 2983 of file cdspop.cc.

References dsp_filter_t.

6.7.2.433 void dsp_filter_design_br2f (dsp_filter_t, float *, float *, float)

Definition at line 2990 of file cdspop.cc.

References dsp_filter_t.

6.7.2.434 void dsp_filter_design_br2 (dsp_filter_t, double *, double *, double)

Definition at line 2998 of file cdspop.cc.

References dsp_filter_t.

6.7.2.435 dsp_flip_t dsp_flip_new (void)

Creates new instance of band flipping object.

Returns:

FlipBand instance

Definition at line 3009 of file cdspop.cc.

References dsp_flip_t.

6.7.2.436 void dsp_flip_delete (dsp_flip_t)

Deletes instance of band flipping object.

Parameters:

dspflipInst FlipBand instance

Definition at line 3015 of file cdspop.cc.

References dsp_flip_t.

6.7.2.437 void dsp_flip_initf (dsp_flip_t, long)

clFlipBand::Initialize

Definition at line 3021 of file cdspop.cc.

References dsp_flip_t.

6.7.2.438 void dsp_flip_init (dsp_flip_t, long)

Definition at line 3029 of file cdspop.cc.

References dsp_flip_t.

6.7.2.439 void dsp_flip_uninit (dsp_flip_t)

clFlipBand::Uninitialize

Definition at line 3037 of file cdspop.cc.

References dsp_flip_t.

6.7.2.440 void dsp_flip_putf (dsp_flip_t, const float *, long)

clFlipBand::Put

Definition at line 3043 of file cdspop.cc.

References dsp_flip_t.

6.7.2.441 void dsp_flip_put (dsp_flip_t, const double *, long)

Definition at line 3049 of file cdspop.cc.

References dsp_flip_t.

6.7.2.442 void dsp_flip_getf (dsp_flip_t, float *, long)

clFlipBand::Get

Definition at line 3055 of file cdspop.cc.

References dsp_flip_t.

6.7.2.443 void dsp_flip_get (dsp_flip_t, double *, long)

Definition at line 3061 of file cdspop.cc.

References dsp_flip_t.

6.7.2.444 void dsp_flip_clear (dsp_flip_t)

clFlipBand::Clear

Definition at line 3067 of file cdspop.cc.

References dsp_flip_t.

6.7.2.445 dsp_rebuf_t dsp_rebuf_new (void)

Creates new instance of rebuffering object.

Returns:

Rebuffer instance

Definition at line 3076 of file cdspop.cc.

References dsp_rebuf_t.

6.7.2.446 void dsp_rebuf_delete (dsp_rebuf_t)

Deletes instance of rebuffering object.

Parameters:

dsprebufInst Rebuffer instance

Definition at line 3082 of file cdspop.cc.

References dsp_rebuf_t.

6.7.2.447 void dsp_rebuf_putf (dsp_rebuf_t, const float ∗, long)

clReBuffer::Put

Definition at line 3088 of file cdspop.cc.

References dsp_rebuf_t.

6.7.2.448 void dsp_rebuf_put (dsp_rebuf_t, const double ∗, long)

Definition at line 3095 of file cdspop.cc.

References dsp_rebuf_t.

6.7.2.449 int dsp_rebuf_getf (dsp_rebuf_t, float ∗, long)

clReBuffer::Get

Definition at line 3102 of file cdspop.cc.

References dsp_rebuf_t.

6.7.2.450 int dsp_rebuf_get (dsp_rebuf_t, double ∗, long)

Definition at line 3111 of file cdspop.cc.

References dsp_rebuf_t.

6.7.2.451 long dsp_rebuf_size (dsp_rebuf_t)

clReBuffer::GetCount

Definition at line 3120 of file cdspop.cc.

References dsp_rebuf_t.

6.7.2.452 void dsp_rebuf_clear (dsp_rebuf_t)

clReBuffer::Clear

Definition at line 3126 of file cdspop.cc.

References dsp_rebuf_t.

6.7.2.453 void dsp_rebuf_copy (dsp_rebuf_t, dsp_rebuf_t)

clReBuffer::operator=

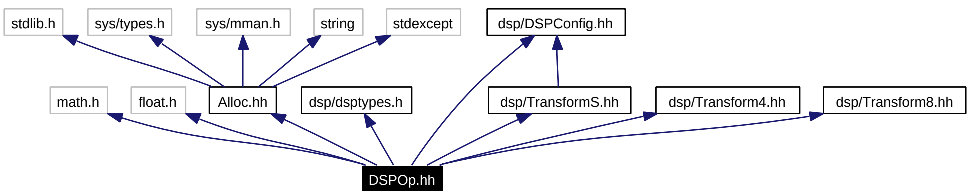
Definition at line 3132 of file cdspop.cc.

References dsp_rebuf_t.

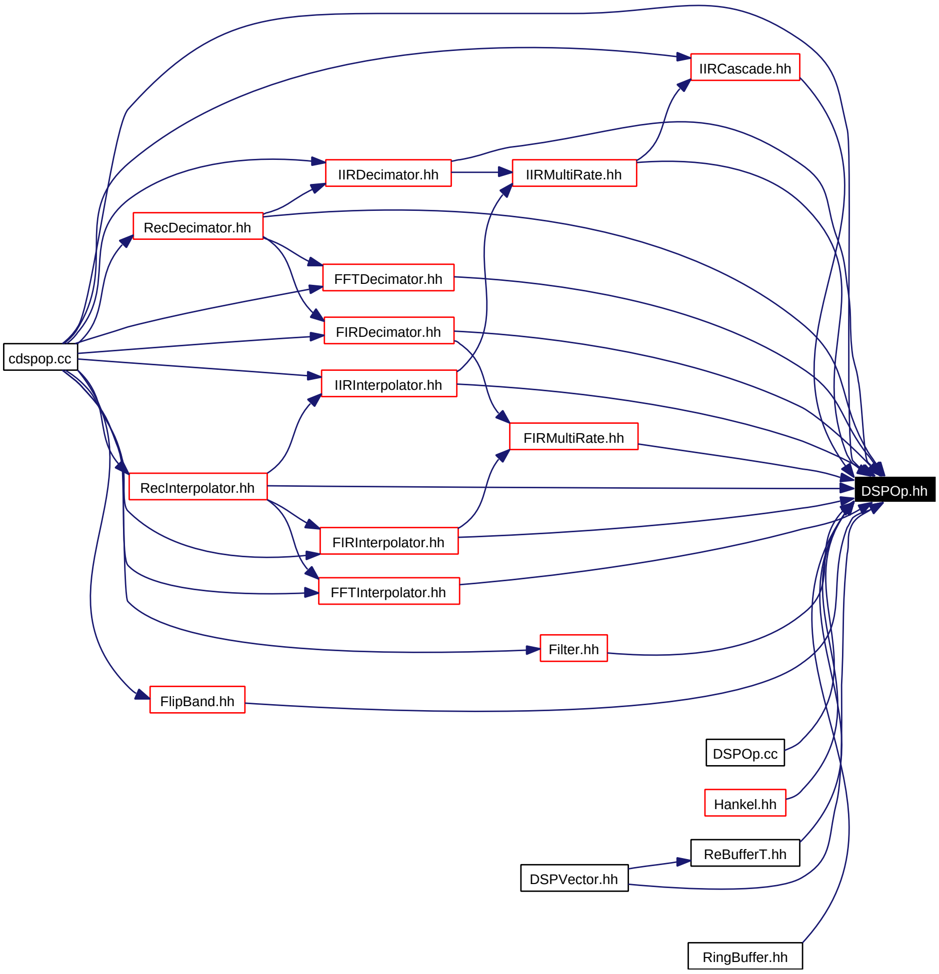
6.8 DSPOp.hh File Reference

```
#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"
```

Include dependency graph for DSPOp.hh:



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



Classes

- class clDSPAlloc

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

- class clDSPOp

    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

Definition at line 50 of file DSPOp.hh.

6.8.1.2 #define INLINE inline

Definition at line 55 of file DSPOp.hh.

Referenced by clDSPOp::Cart2Polar(), clDSPOp::CplxAdd(), clDSPOp::CplxConj(), clDSPOp::CplxDiv(), clDSPOp::CplxExp(), clDSPOp::CplxLog(), clDSPOp::Cplx-Log10(), clDSPOp::CplxMul(), clDSPOp::CplxMulC(), clDSPOp::CplxPow(), clDSPOp::CplxRoot(), clDSPOp::CplxSub(), DoubleCompare(), FloatCompare(), LongCompare(), clDSPOp::ModZeroBessel(), clDSPOp::Multiple(), and cl-DSPOp::Polar2Cart().

6.8.1.3 #define DSP_MAXBESSEL 32L

Definition at line 114 of file DSPOp.hh.

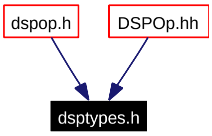
Referenced by clDSPOp::ModZeroBessel().

6.8.1.4 #define DSP_WISDOM_FILE "fftw.wisdom"

Definition at line 117 of file DSPOp.hh.

6.9 dsptypes.h File Reference

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



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.

Referenced by clDSPOp::Add(), clDSPOp::Copy(), clDSPOp::CplxPow(), clDSPOp::CplxRoot(), clDSPOp::Div(), dsp_caddf(), dsp_cdivf(), dsp_cmulf(), dsp_cset2f(), dsp_csetf(), dsp_csubf(), clHankel::Initialize(), clFlipBand::Initialize(), clFilter::Initialize(), clDSPOp::Mul(), clHankel::Process0(), clHankel::Process1(), clDSPOp::Reverse(), clDSPOp::Set(), and clDSPOp::Sub().

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.

Referenced by clDSPOp::Add(), clDSPOp::Cart2Polar(), clDSPOp::CartToPolar(), clDSPOp::Copy(), clDSPOp::CplxAdd(), clDSPOp::CplxConj(), clDSPOp::CplxDiv(), clDSPOp::CplxExp(), clDSPOp::CplxLog(), clDSPOp::CplxLog10(), clDSPOp::CplxMul(), clDSPOp::CplxMulC(), clDSPOp::CplxPow(), clDSPOp::CplxRoot(), clDSPOp::CplxSub(), clDSPOp::Div(), dsp_cadd2f(), dsp_cadd3f(), dsp_caddf(), dsp_cart2polar2f(), dsp_cart2polar3f(), dsp_ccmulf(), dsp_ccmulf_nip(), dsp_ccopyf(), dsp_cdiv2f(), dsp_cdiv3f(), dsp_cdivf(), dsp_cfftf_nip(), dsp_chmul2f(), dsp_chmulf(), dsp_cifftf_nip(), dsp_cmul2f(), dsp_cmul3f(), dsp_cmulf(), dsp_cset2f(), dsp_csetf(), dsp_csub2f(), dsp_csub3f(), dsp_csubf(), dsp_czerof(), dsp_fftf(), dsp_fftf_nip(), dsp_fftw_convertcf2d(), dsp_fftw_convertcf2f(), dsp_fftw_convertd2cf(), dsp_fftw_convertf2cf(), dsp_filter_get_ccoeffsf(), dsp_filter_put2f(), dsp_filter_set_ccoeffsf(), dsp_ifftf_nip(), dsp_magnitudef(), dsp_phasef(), dsp_polar2cart2f(), dsp_polar2cart3f(), dsp_powerf(), dsp_powerphasef(), clDSPOp::FFTi(), clDSPOp::FFTo(), clDSPOp::FFTWConvert(), clFilter::GetCoeffs(), clDSPOp::IFFTo(), clFilter::InitCoeffsS(), clDSPOp::Magnitude(), clDSPOp::Mul(), clDSPOp::MulC(), clDSPAlloc::operator stSCplx *(), clDSPOp::Phase(), clDSPOp::Polar2Cart(), clDSPOp::PolarToCart(), clDSPOp::Power(), clDSPOp::PowerPhase(), clHankel::Process0(), clHankel::Process1(), clFlipBand::Put(), clFilter::Put(), clFilter::ReadyFilterS(), clDSPOp::Reverse(), clDSPOp::Set(), clFilter::SetCoeffs(), clDSPOp::Sub(), and clDSPOp::Zero().

6.9.1.3 typedef struct _sDCplx stDCplx

Double precision complex datatype.

Referenced by clDSPOp::Add(), clDSPOp::Copy(), clDSPOp::CplxPow(), clDSPOp::CplxRoot(), clDSPOp::Div(), dsp_cadd(), dsp_cdiv(), dsp_cmul(), dsp_cset(), dsp_cset2(), dsp_csub(), clHankel::Initialize(), clFlipBand::Initialize(), clFilter::Initialize(), clDSPOp::Mul(), clHankel::Process0(), clHankel::Process1(), clDSPOp::Reverse(), clDSPOp::Set(), and clDSPOp::Sub().

6.9.1.4 typedef struct _sDCplx * stpDCplx

Double precision complex datatype.

Referenced by clDSPOp::Add(), clDSPOp::Cart2Polar(), clDSPOp::CartToPolar(), clDSPOp::Copy(), clDSPOp::CplxAdd(), clDSPOp::CplxConj(), clDSPOp::CplxDiv(), clDSPOp::CplxExp(), clDSPOp::CplxLog(), clDSPOp::CplxLog10(), clDSPOp::CplxMul(), clDSPOp::CplxMulC(), clDSPOp::CplxPow(), clDSPOp::CplxRoot(), clDSPOp::CplxSub(), clDSPOp::Div(), dsp_cadd(), dsp_cadd2(), dsp_cadd3(), dsp_cart2polar2(), dsp_cart2polar3(), dsp_ccmul(), dsp_ccmul_nip(), dsp_ccopy(), dsp_cdiv(), dsp_cdiv2(), dsp_cdiv3(), dsp_cfft_nip(), dsp_chmul(), dsp_chmul2(), dsp_cifft_nip(), dsp_cmul(), dsp_cmul2(), dsp_cmul3(), dsp_cset(), dsp_cset2(), dsp_csub(), dsp_csub2(), dsp_csub3(), dsp_czero(), dsp_fft(), dsp_-fft_nip(), dsp_fftw_convertcd2d(), dsp_fftw_convertcd2f(), dsp_fftw_convertd2cd(), dsp_fftw_convertf2cd(), dsp_filter_get_ccoeffs(), dsp_filter_put2(), dsp_filter_set_-ccoeffs(), dsp_ifft_nip(), dsp_magnitude(), dsp_phase(), dsp_polar2cart2(), dsp_-polar2cart3(), dsp_power(), dsp_powerphase(), clDSPOp::FFTi(), clDSPOp::FFTo(), clDSPOp::FFTWConvert(), clFilter::GetCoeffs(), clDSPOp::IFFTo(), clFilter::Init-CoeffsD(), clDSPOp::Magnitude(), clDSPOp::Mul(), clDSPOp::MulC(), cl-DSPAlloc::operator stDCplx ∗(), clDSPOp::Phase(), clDSPOp::Polar2Cart(), clDSPOp::PolarToCart(), clDSPOp::Power(), clDSPOp::PowerPhase(), cl-Hankel::Process0(), clHankel::Process1(), clFlipBand::Put(), clFilter::Put(), cl-Filter::ReadyFilterD(), clDSPOp::Reverse(), clDSPOp::Set(), clFilter::SetCoeffs(), clDSPOp::Sub(), and clDSPOp::Zero().

6.9.1.5 typedef struct _sSPolar stSPolar

Single precision polar datatype.

6.9.1.6 typedef struct _sSPolar ∗ stpSPolar

Single precision polar datatype.

Referenced by clDSPOp::Cart2Polar(), clDSPOp::CartToPolar(), dsp_cart2polar3f(), dsp_polar2cart3f(), clDSPAlloc::operator stSPolar ∗(), clDSPOp::Polar2Cart(), and cl-DSPOp::PolarToCart().

6.9.1.7 typedef struct _sDPolar stDPolar

Double precision polar datatype.

6.9.1.8 typedef struct _sDPolar ∗ stpDPolar

Double precision polar datatype.

Referenced by clDSPOp::Cart2Polar(), clDSPOp::CartToPolar(), dsp_cart2polar3(), dsp_polar2cart3(), clDSPAlloc::operator stDPolar ∗(), clDSPOp::Polar2Cart(), and cl-DSPOp::PolarToCart().

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.

Referenced by clDSPOp::Cart2Polar(), clDSPOp::CartToPolar(), dsp_cart2polar4f(), dsp_polar2cart4f(), clDSPOp::Polar2Cart(), and clDSPOp::PolarToCart().

6.9.1.11 typedef union _uDCoord utDCoord

Double precision cartesian/polar datatype.

6.9.1.12 typedef union _uDCoord * utpDCoord

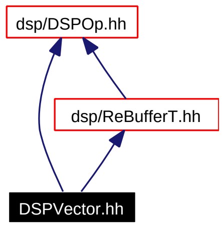
Double precision cartesian/polar datatype.

Referenced by clDSPOp::Cart2Polar(), clDSPOp::CartToPolar(), dsp_cart2polar4(), dsp_polar2cart4(), clDSPOp::Polar2Cart(), and clDSPOp::PolarToCart().

6.10 DSPVector.hh File Reference

```
#include "dsp/DSPOp.hh"
#include "dsp/ReBufferT.hh"
```

Include dependency graph for DSPVector.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())

Definition at line 30 of file DSPVector.hh.

Referenced by clDSPVector< TDSPVector_t >::AutoCorr(), clDSPVector< TDSPVector_t >::Conv(), clDSPVector< TDSPVector_t >::Corr(), clDSPVector< TDSPVector_t >::CrossCorr(), clDSPVector< TDSPVector_t >::DotProduct(), clDSPVector< TDSPVector_t >::FIRFilter(), clDSPVector< TDSPVector_t >::Magnitude(), clDSPVector< TDSPVector_t >::MulC(), clDSPVector< TDSPVector_t >::operator *(), clDSPVector< TDSPVector_t >::operator *=(), clDSPVector< TDSPVector_t >::operator+(), clDSPVector< TDSPVector_t >::operator+=(), clDSPVector< TDSPVector_t >::operator-(), clDSPVector< TDSPVector_t >::operator-=(), clDSPVector< TDSPVector_t >::operator/(), clDSPVector< TDSPVector_t >::operator/=(), clDSPVector< TDSPVector_t

>::Phase(), clDSPVector< TDSPVector_t >::Power(), clDSPVector< TDSPVector_t >::Reverse(), and clDSPVector< TDSPVector_t >::Scale().

6.10.1.2 #define DSPV_SHORTER2(x, y) ((x.Size() <= y.Size()) ? x.Size() : y.Size())

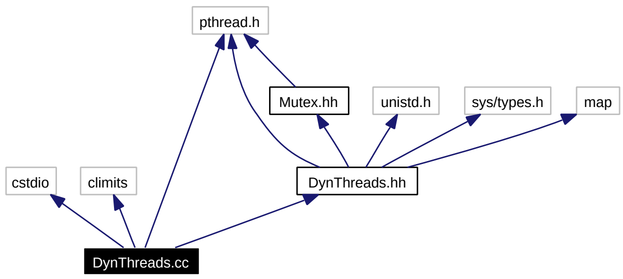
Definition at line 31 of file DSPVector.hh.

Referenced by clDSPVector< TDSPVector_t >::Conv(), clDSPVector< TDSPVector_t >::Corr(), and clDSPVector< TDSPVector_t >::CrossCorr().

6.11 DynThreads.cc File Reference

```
#include <cstdio>
#include <climits>
#include <pthread.h>
#include "DynThreads.hh"
```

Include dependency graph for DynThreads.cc:



Functions

- void ∗ WrapDynThreadBase (void ∗vpParam)

6.11.1 Function Documentation

6.11.1.1 void∗ WrapDynThreadBase (void ∗ *vpParam*)

Definition at line 37 of file DynThreads.cc.

References clDynThreadsBase::InternalCaller(), clDynThreadsBase::_stParams::Klass, clDynThreadsBase::stpParams, and clDynThreadsBase::_stParams::vpParam.

Referenced by clDynThreadsBase::Create().

6.12 DynThreads.hh File Reference

`#include <unistd.h>`

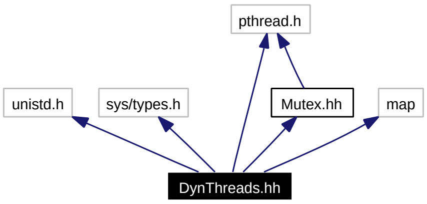
`#include <sys/types.h>`

`#include <pthread.h>`

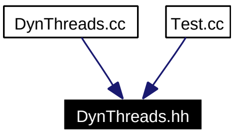
`#include <map>`

`#include <Mutex.hh>`

Include dependency graph for DynThreads.hh:



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



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

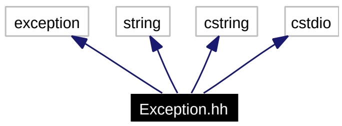
6.12.1.1 typedef std::map<int, pthread_t> ThreadMap_t

Definition at line 39 of file DynThreads.hh.

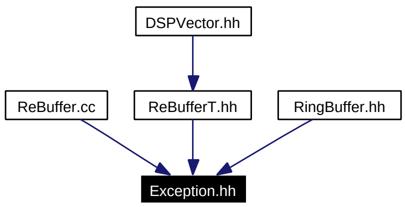
6.13 Exception.hh File Reference

```
#include <exception>
#include <string>
#include <cstring>
#include <cstdio>
```

Include dependency graph for Exception.hh:



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



Classes

- class clException

6.14 FFTDecimator.cc File Reference

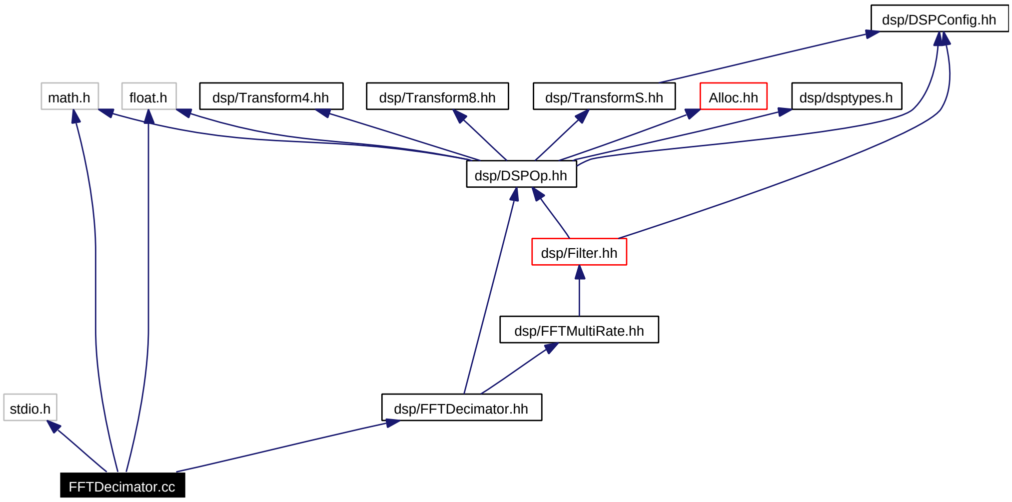
`#include <stdio.h>`

`#include <math.h>`

`#include <float.h>`

`#include "dsp/FFTDecimator.hh"`

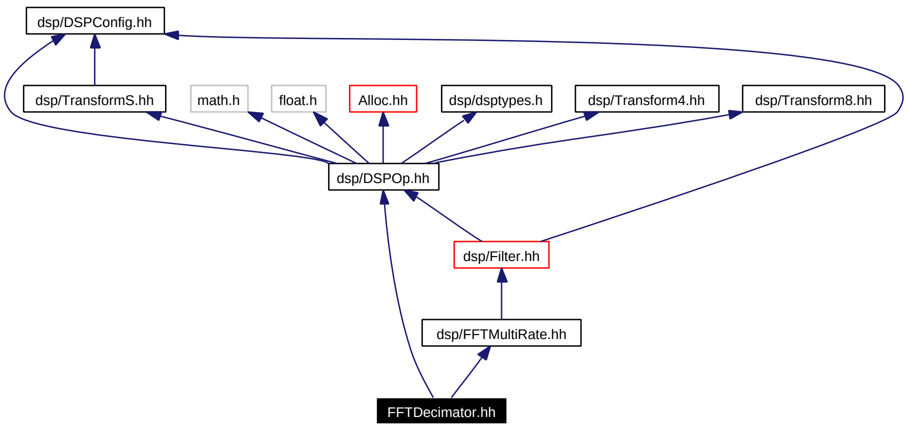
Include dependency graph for FFTDecimator.cc:



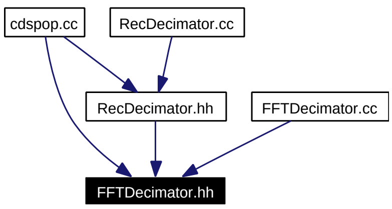
6.15 FFTDecimator.hh File Reference

```
#include "dsp/DSPOp.hh"
#include "dsp/FFTMultiRate.hh"
```

Include dependency graph for FFTDecimator.hh:



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



Classes

- class clFFTDecimator

  FFT decimation filter class implementation.

6.16 FFTInterpolator.cc File Reference

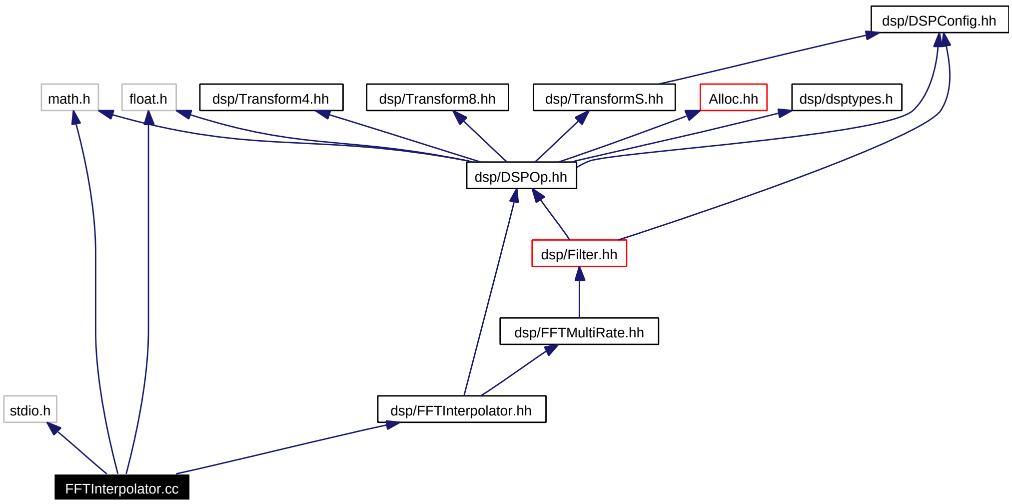
`#include <stdio.h>`

`#include <math.h>`

`#include <float.h>`

`#include "dsp/FFTInterpolator.hh"`

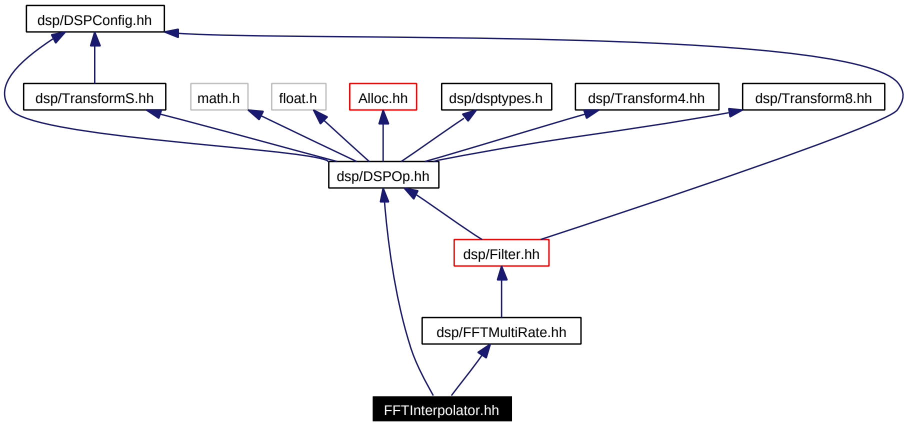
Include dependency graph for FFTInterpolator.cc:



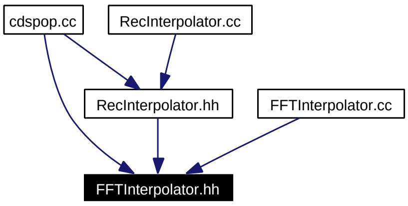
6.17 FFTInterpolator.hh File Reference

```
#include "dsp/DSPOp.hh"
#include "dsp/FFTMultiRate.hh"
```

Include dependency graph for FFTInterpolator.hh:



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



Classes

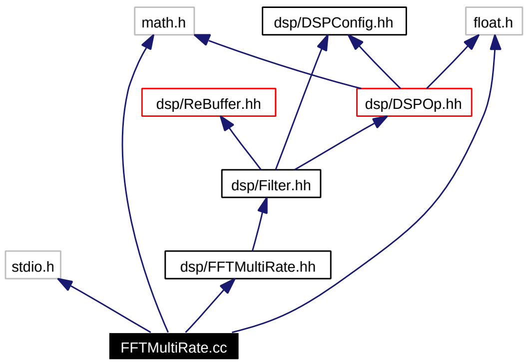
- class clFFTInterpolator

  FFT interpolation filter class implementation.

6.18 FFTMultiRate.cc File Reference

```
#include <stdio.h>
#include <math.h>
#include <float.h>
#include "dsp/FFTMultiRate.hh"
```

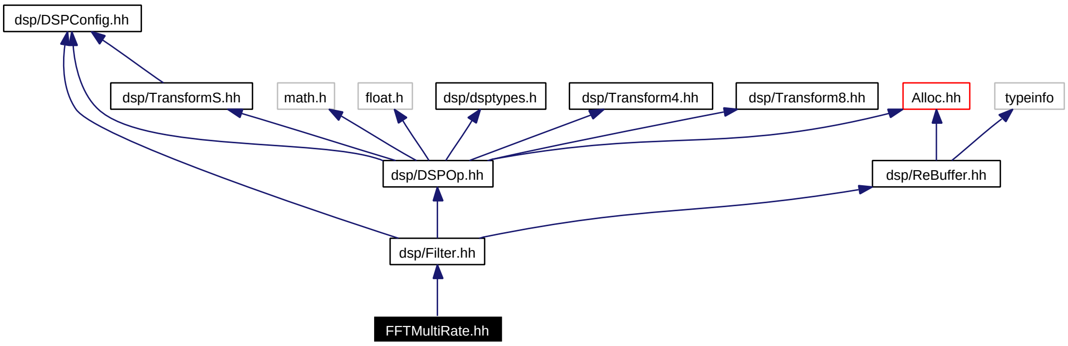
Include dependency graph for FFTMultiRate.cc:



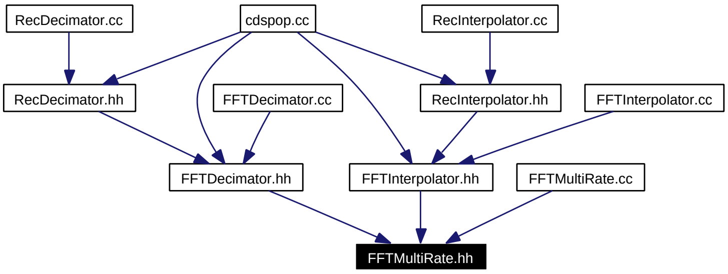
6.19 FFTMultiRate.hh File Reference

`#include "dsp/Filter.hh"`

Include dependency graph for FFTMultiRate.hh:



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



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

Definition at line 29 of file FFTMultiRate.hh.

Referenced by clFFTMultiRate::Initialize().

6.19.1.2 #define FFTMULTIRATE_OVERLAP 0.75f

Definition at line 30 of file FFTMultiRate.hh.

Referenced by clFFTMultiRate::Initialize().

6.19.1.3 #define FFTMULTIRATE_DELTAOMEGA 0.1f

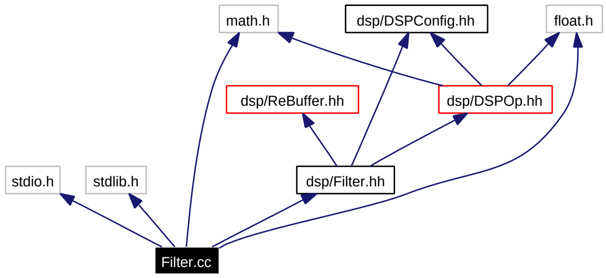
Definition at line 31 of file FFTMultiRate.hh.

Referenced by clFFTMultiRate::Initialize().

6.20 Filter.cc File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <float.h>
#include "dsp/Filter.hh"
```

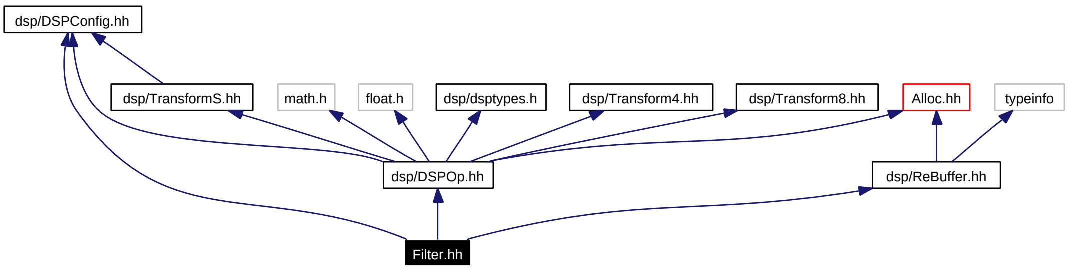
Include dependency graph for Filter.cc:



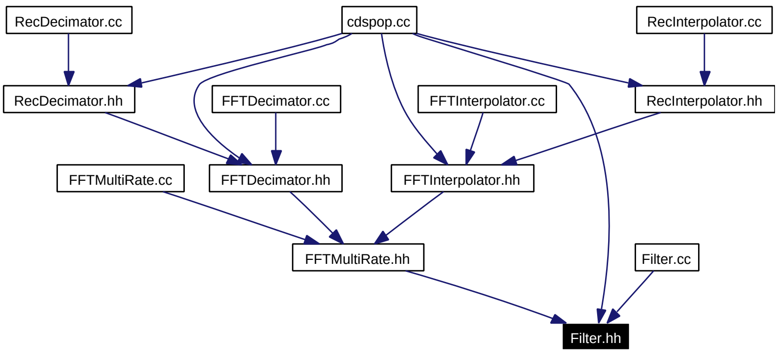
6.21 Filter.hh File Reference

```
#include "dsp/DSPConfig.hh"
#include "dsp/DSPOp.hh"
#include "dsp/ReBuffer.hh"
```

Include dependency graph for Filter.hh:



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



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

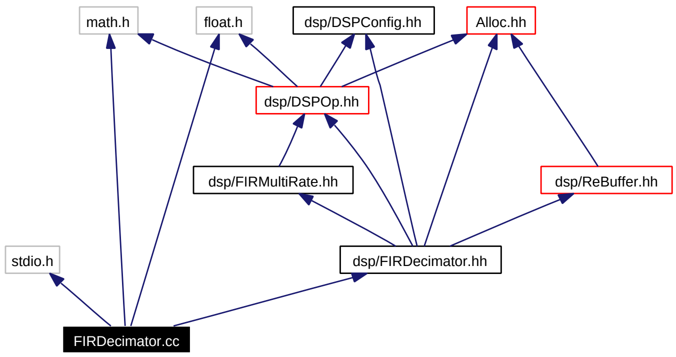
FILTER_SMOOTH_DOLPH_CHEBYSHEV

Definition at line 35 of file Filter.hh.

6.22 FIRDecimator.cc File Reference

```
#include <stdio.h>
#include <math.h>
#include <float.h>
#include "dsp/FIRDecimator.hh"
```

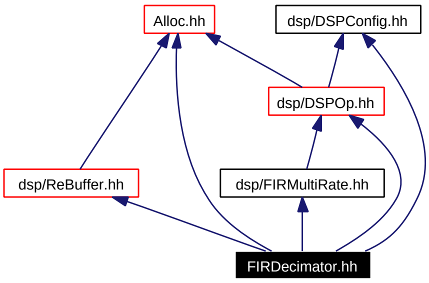
Include dependency graph for FIRDecimator.cc:



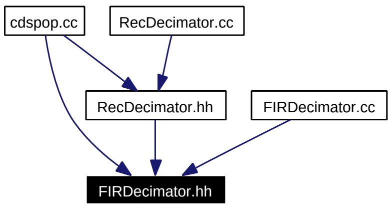
6.23 FIRDecimator.hh File Reference

```
#include <Alloc.hh>
#include "dsp/DSPConfig.hh"
#include "dsp/DSPOp.hh"
#include "dsp/ReBuffer.hh"
#include "dsp/FIRMultiRate.hh"
```

Include dependency graph for FIRDecimator.hh:



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



Classes

- class clFIRDecimator

  FIR decimation filter class implementation.

6.24 FIRInterpolator.cc File Reference

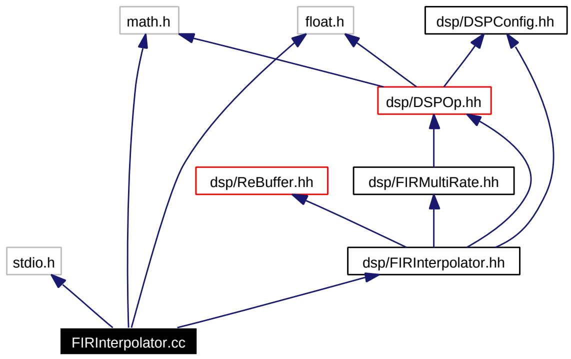
`#include <stdio.h>`

`#include <math.h>`

`#include <float.h>`

`#include "dsp/FIRInterpolator.hh"`

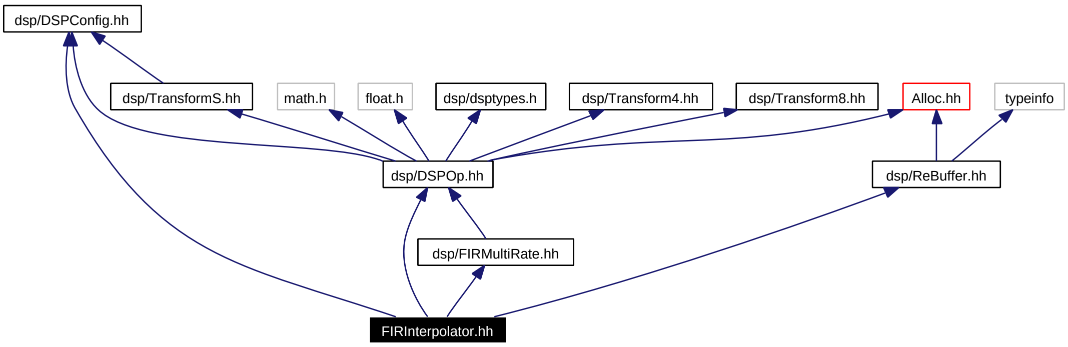
Include dependency graph for FIRInterpolator.cc:



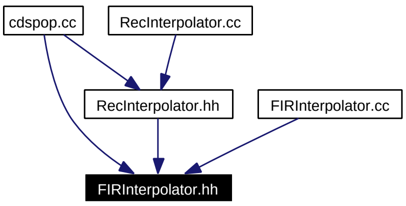
6.25 FIRInterpolator.hh File Reference

```
#include "dsp/DSPConfig.hh"
#include "dsp/DSPOp.hh"
#include "dsp/ReBuffer.hh"
#include "dsp/FIRMultiRate.hh"
```

Include dependency graph for FIRInterpolator.hh:



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



Classes

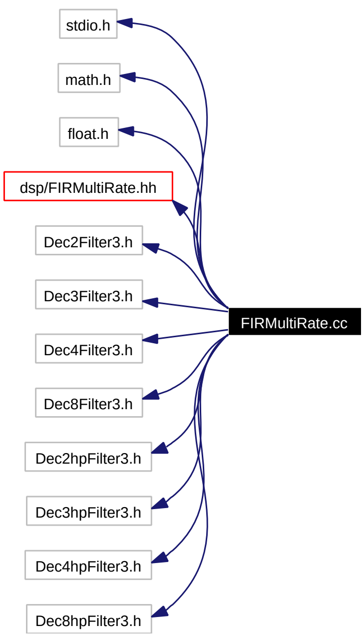
- class clFIRInterpolator

  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"
```

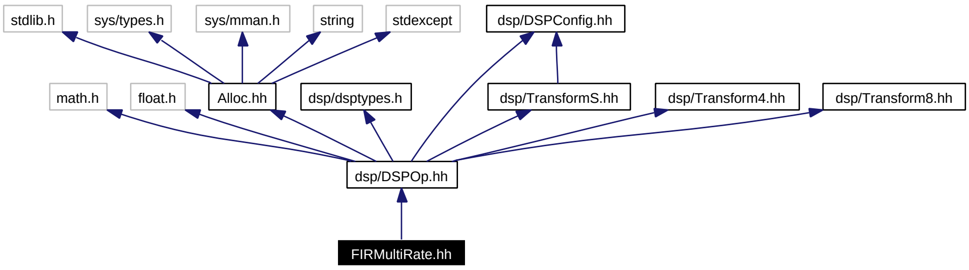
Include dependency graph for FIRMultiRate.cc:



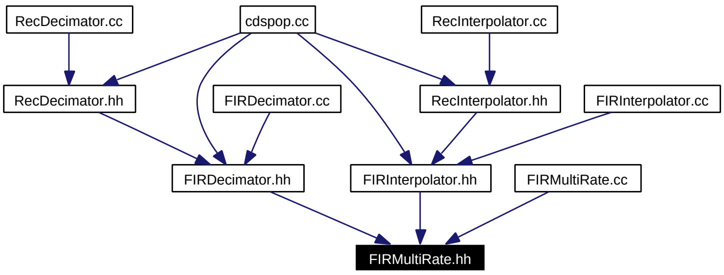
6.27 FIRMultiRate.hh File Reference

```
#include "dsp/DSPOp.hh"
```

Include dependency graph for FIRMultiRate.hh:



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



Classes

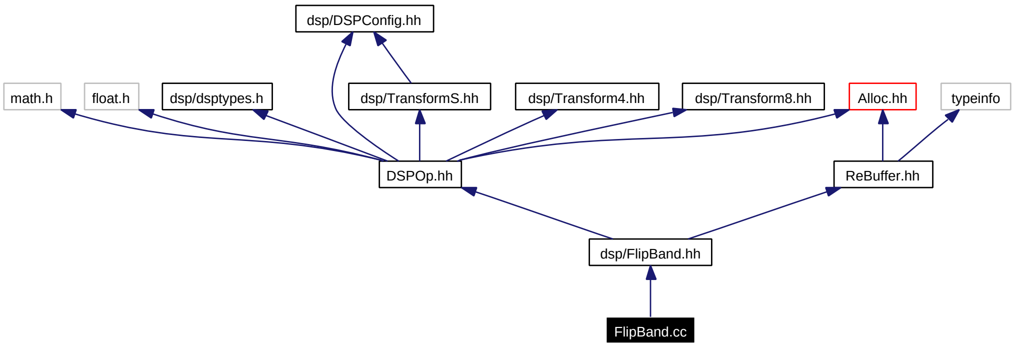
- class clFIRMultiRate

    Base class for FIR based multirate filters.

6.28 FlipBand.cc File Reference

`#include "dsp/FlipBand.hh"`

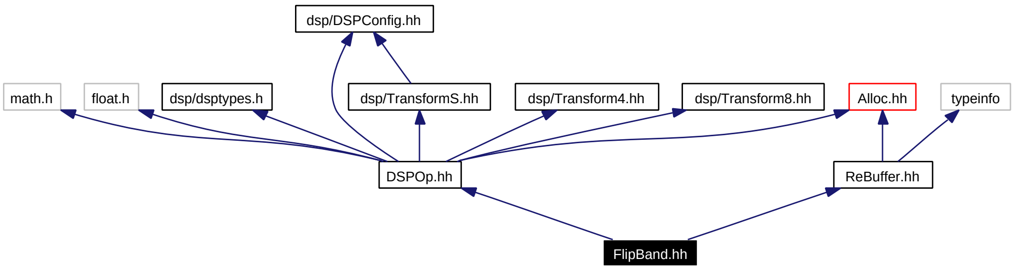
Include dependency graph for FlipBand.cc:



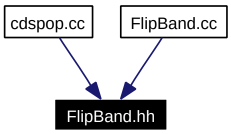
6.29 FlipBand.hh File Reference

```
#include "DSPOp.hh"
#include "ReBuffer.hh"
```

Include dependency graph for FlipBand.hh:



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



Classes

- class clFlipBand

6.30 Generic.hh File Reference

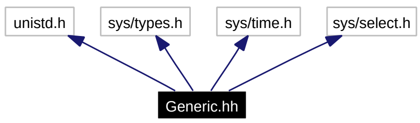
`#include <unistd.h>`

`#include <sys/types.h>`

`#include <sys/time.h>`

`#include <sys/select.h>`

Include dependency graph for Generic.hh:



Functions

- void uSleep (long lUSecs)

6.30.1 Function Documentation

6.30.1.1 void uSleep (long *lUSecs*) `[inline]`

Definition at line 33 of file Generic.hh.

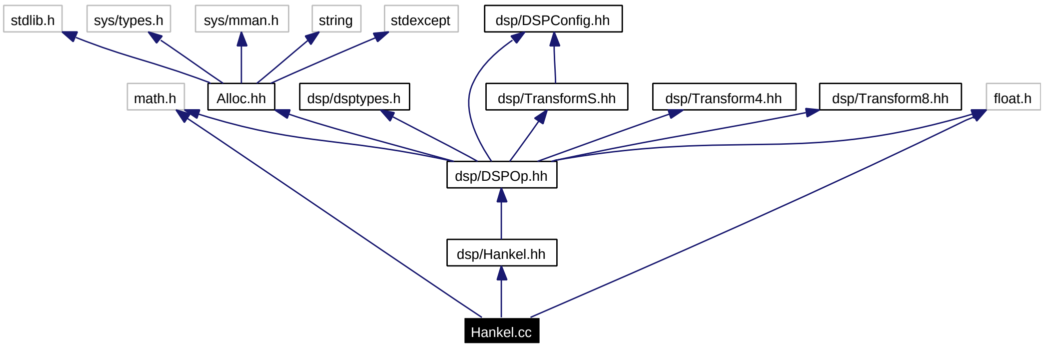
6.31 Hankel.cc File Reference

`#include <math.h>`

`#include <float.h>`

`#include "dsp/Hankel.hh"`

Include dependency graph for Hankel.cc:



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]

6.31.1 Define Documentation

6.31.1.1 #define ABEL_NSE 9

Definition at line 42 of file Hankel.cc.

Referenced by clHankel::DoAbel(), and clHankel::InitAbel().

6.31.2 Variable Documentation

6.31.2.1 const float fpH[ABEL_NSE]

Initial value:

```
{
    1.000000000000000000f,
    0.610926299405048390f,
    0.895089852938535935f,
    1.34082948787002865f,
    2.02532848558443890f,
    3.18110895533701843f,
    5.90898360396353794f,
    77.6000213494180286f,
    528.221800846070892f
}
```

Definition at line 43 of file Hankel.cc.

Referenced by clHankel::InitAbel().

6.31.2.2 const double dpH[ABEL_NSE]

Initial value:

```
{
    1.000000000000000000,
    0.610926299405048390,
    0.895089852938535935,
    1.34082948787002865,
    2.02532848558443890,
    3.18110895533701843,
    5.90898360396353794,
    77.6000213494180286,
    528.221800846070892
}
```

Definition at line 54 of file Hankel.cc.

6.31.2.3 const float fpLambda[ABEL_NSE]

Initial value:

```
{
    0.000000000000000000f,
    -2.08424632126539366f,
    -5.78928630565552371f,
    -14.6268676854951032f,
    -35.0617158334443104f,
    -83.3258406398958158f,
```

```
    -210.358805421311445f,
    -6673.64911325382036f,
    -34897.7050244132261f
}
```

Definition at line 65 of file Hankel.cc.

Referenced by clHankel::InitAbel().

6.31.2.4 const double dpLambda[ABEL_NSE]

Initial value:

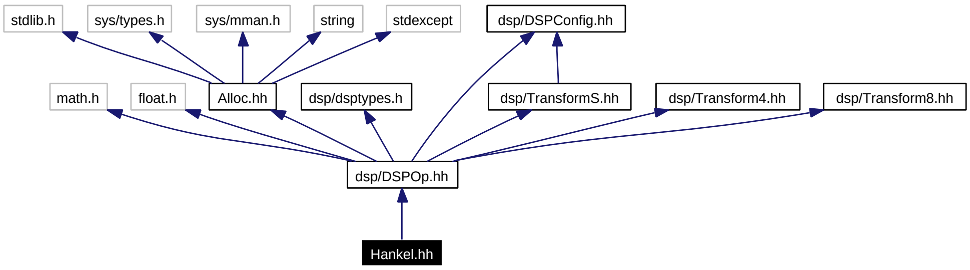
```
 {
    0.000000000000000000,
    -2.08424632126539366,
    -5.78928630565552371,
    -14.6268676854951032,
    -35.0617158334443104,
    -83.3258406398958158,
    -210.358805421311445,
    -6673.64911325382036,
    -34897.7050244132261
}
```

Definition at line 76 of file Hankel.cc.

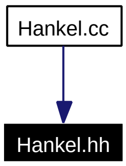
6.32 Hankel.hh File Reference

`#include <dsp/DSPOp.hh>`

Include dependency graph for Hankel.hh:



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



Classes

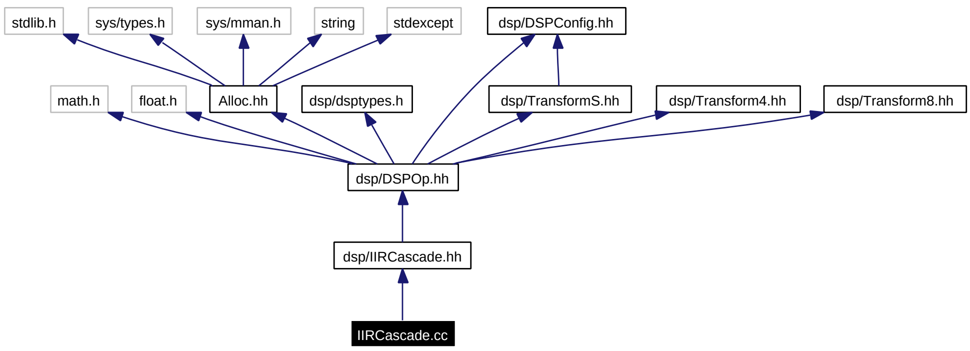
- class clHankel

    Class implementation of (modified) Hankel-transform.

6.33 IIRCascade.cc File Reference

`#include "dsp/IIRCascade.hh"`

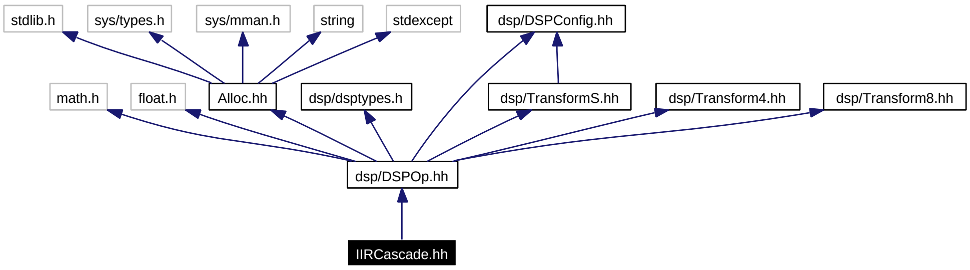
Include dependency graph for IIRCascade.cc:



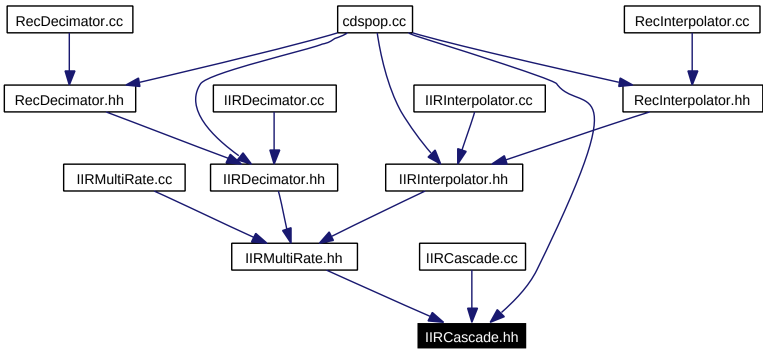
6.34 IIRCascade.hh File Reference

`#include "dsp/DSPOp.hh"`

Include dependency graph for IIRCascade.hh:



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



Classes

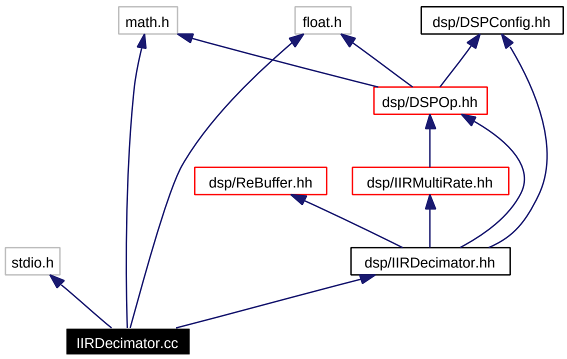
- class clIIRCascade

    Class to handle cascaded IIR stages as one filter.

6.35 IIRDecimator.cc File Reference

```
#include <stdio.h>
#include <math.h>
#include <float.h>
#include "dsp/IIRDecimator.hh"
```

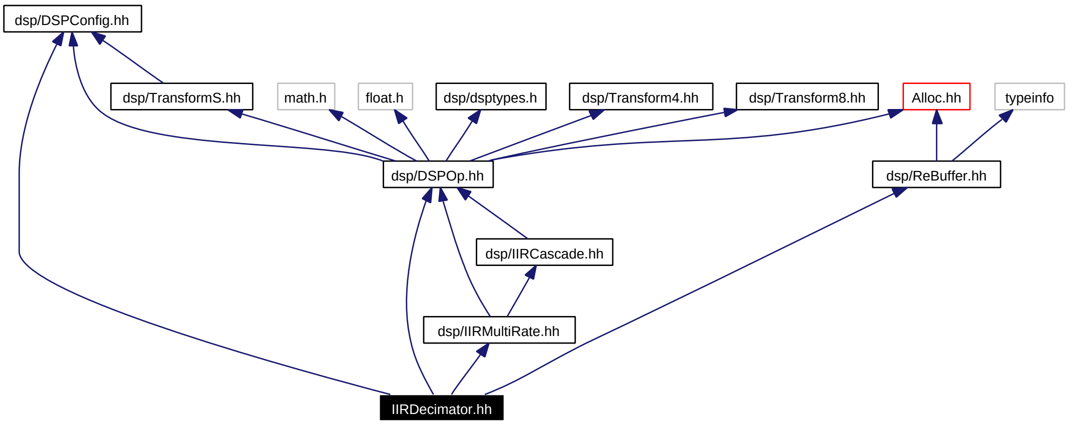
Include dependency graph for IIRDecimator.cc:



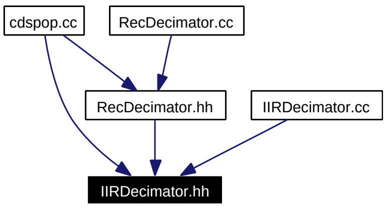
6.36 IIRDecimator.hh File Reference

```
#include "dsp/DSPConfig.hh"
#include "dsp/DSPOp.hh"
#include "dsp/ReBuffer.hh"
#include "dsp/IIRMultiRate.hh"
```

Include dependency graph for IIRDecimator.hh:



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



Classes

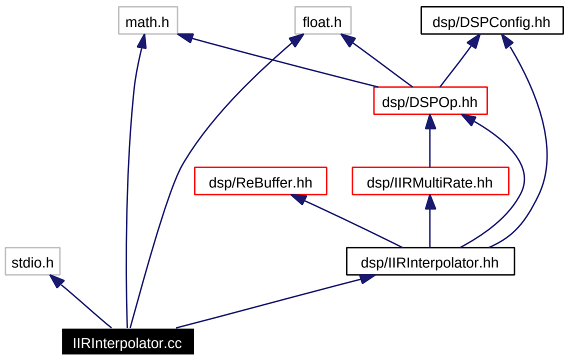
- class clIIRDecimator

  IIR decimation filter class implementation.

6.37 IIRInterpolator.cc File Reference

```
#include <stdio.h>
#include <math.h>
#include <float.h>
#include "dsp/IIRInterpolator.hh"
```

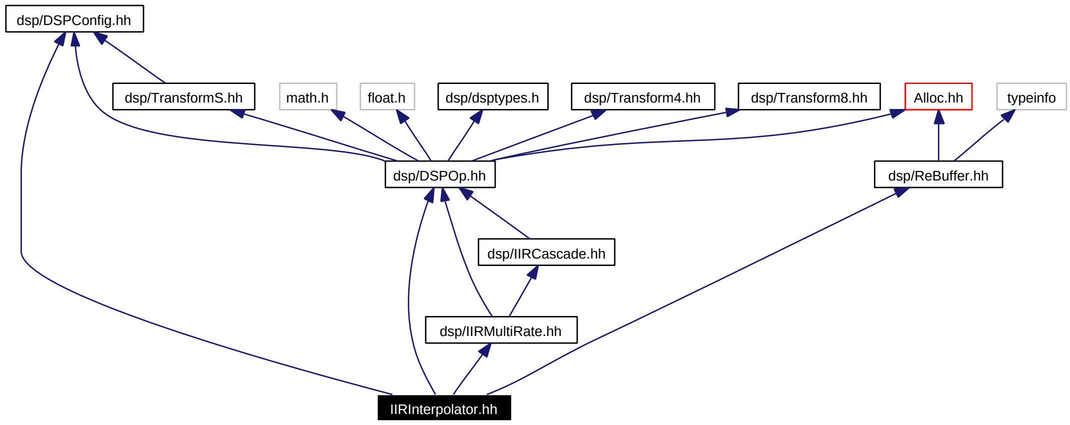
Include dependency graph for IIRInterpolator.cc:



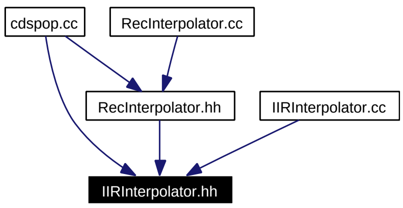
6.38 IIRInterpolator.hh File Reference

```
#include "dsp/DSPConfig.hh"
#include "dsp/DSPOp.hh"
#include "dsp/ReBuffer.hh"
#include "dsp/IIRMultiRate.hh"
```

Include dependency graph for IIRInterpolator.hh:



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



Classes

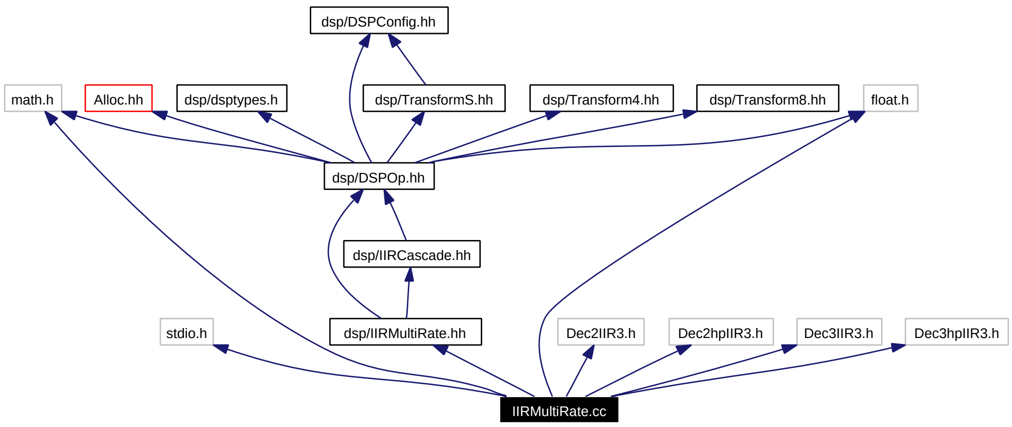
- class clIIRInterpolator

  IIR interpolation filter class implementation.

6.39 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"
```

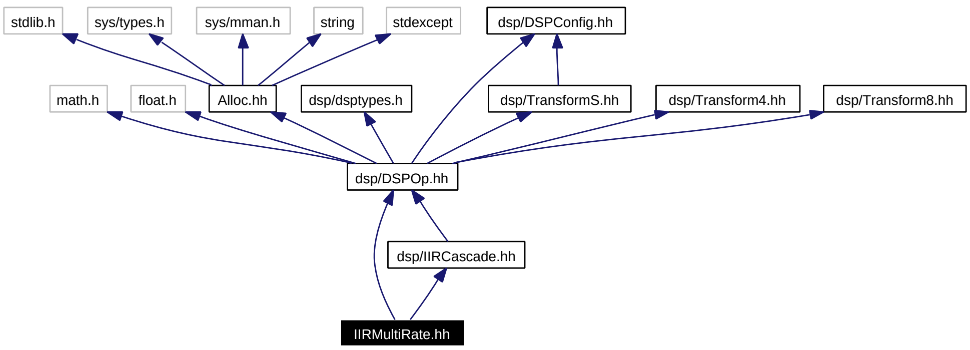
Include dependency graph for IIRMultiRate.cc:



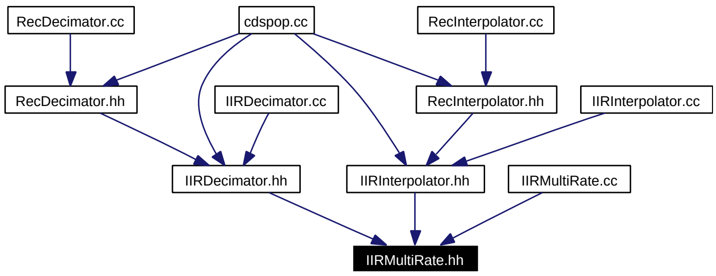
6.40 IIRMultiRate.hh File Reference

```
#include "dsp/DSPOp.hh"
#include "dsp/IIRCascade.hh"
```

Include dependency graph for IIRMultiRate.hh:



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



Classes

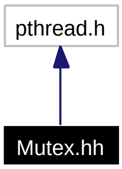
- class clIIRMultiRate

  Base class for IIR based multirate filters.

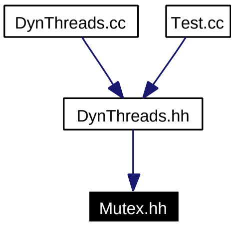
6.41 Mutex.hh File Reference

`#include <pthread.h>`

Include dependency graph for Mutex.hh:



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



Classes

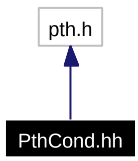
- class clMutex

    Class implementation of POSIX mutex semaphore.

6.42 PthCond.hh File Reference

`#include <pth.h>`

Include dependency graph for PthCond.hh:



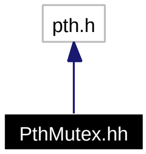
Classes

- class clPthCond

6.43 PthMutex.hh File Reference

```
#include <pth.h>
```

Include dependency graph for PthMutex.hh:



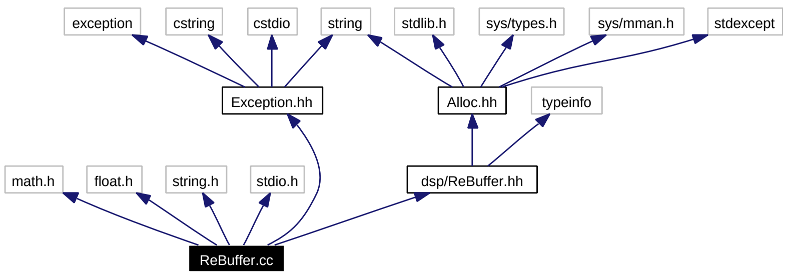
Classes

- class clPthMutex

6.44 ReBuffer.cc File Reference

```
#include <math.h>
#include <float.h>
#include <string.h>
#include <stdio.h>
#include <Exception.hh>
#include "dsp/ReBuffer.hh"
```

Include dependency graph for ReBuffer.cc:

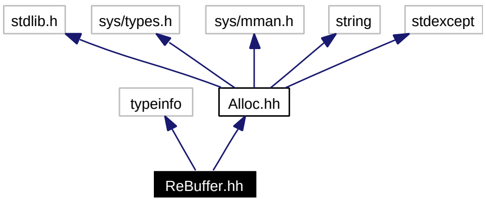


6.45 ReBuffer.hh File Reference

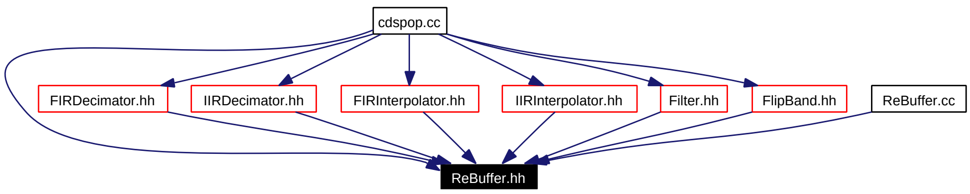
`#include <typeinfo>`

`#include <Alloc.hh>`

Include dependency graph for ReBuffer.hh:



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



Classes

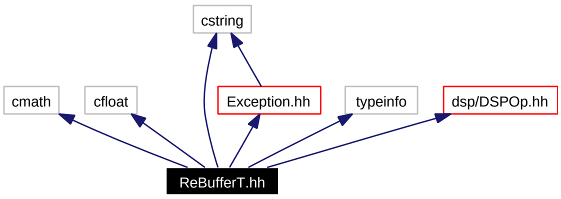
- class clReBuffer

    Class for splitting data into buffers of different sizes.

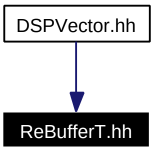
6.46 ReBufferT.hh File Reference

```
#include <cmath>
#include <cfloat>
#include <cstring>
#include <typeinfo>
#include <Exception.hh>
#include "dsp/DSPOp.hh"
```

Include dependency graph for ReBufferT.hh:



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



Classes

- class clXReBufferT

  Exception class for clReBufferT.

- class clReBufferT

  Template class for splitting data into buffers of different sizes.

6.47 RecDecimator.cc File Reference

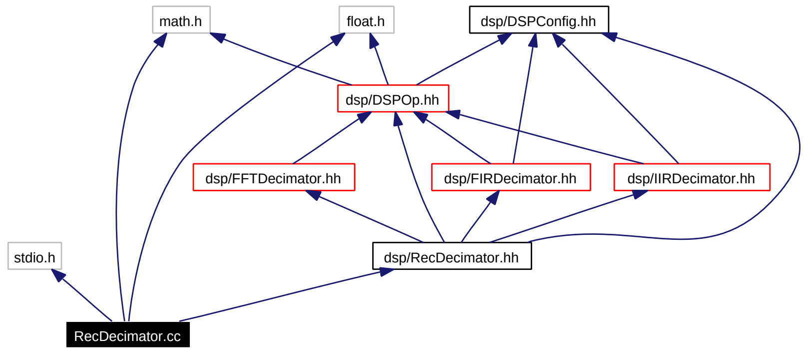
`#include <stdio.h>`

`#include <math.h>`

`#include <float.h>`

`#include "dsp/RecDecimator.hh"`

Include dependency graph for RecDecimator.cc:



6.48 RecDecimator.hh File Reference

`#include "dsp/DSPConfig.hh"`

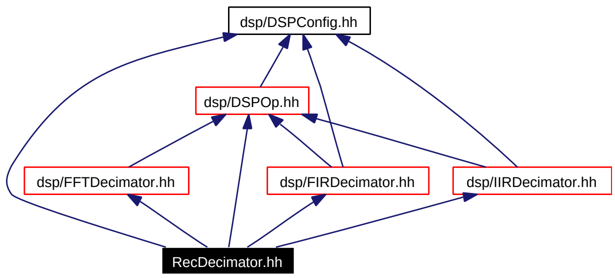
`#include "dsp/DSPOp.hh"`

`#include "dsp/FFTDecimator.hh"`

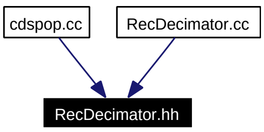
`#include "dsp/FIRDecimator.hh"`

`#include "dsp/IIRDecimator.hh"`

Include dependency graph for RecDecimator.hh:



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



Classes

- class clRecDecimator

    Recursive decimation filter class implementation.

Defines

- #define RECDEC_MAX_SUB_ROUNDS 32

    Maximum number of subsequent rounds.

6.48.1 Define Documentation

6.48.1.1 #define RECDEC_MAX_SUB_ROUNDS 32

Maximum number of subsequent rounds.

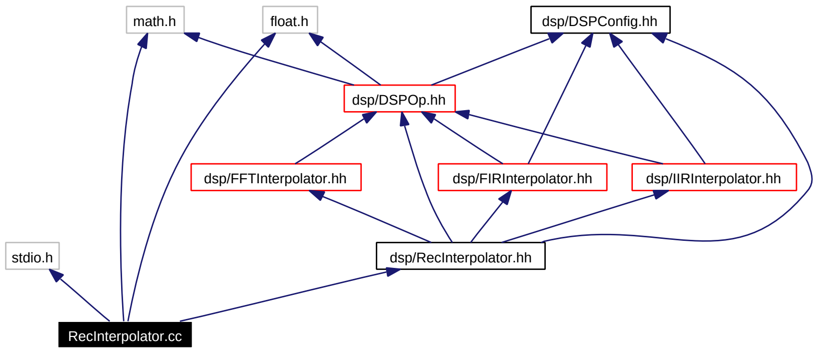
Definition at line 33 of file RecDecimator.hh.

Referenced by clRecDecimator::Initialize().

6.49 RecInterpolator.cc File Reference

```
#include <stdio.h>
#include <math.h>
#include <float.h>
#include "dsp/RecInterpolator.hh"
```

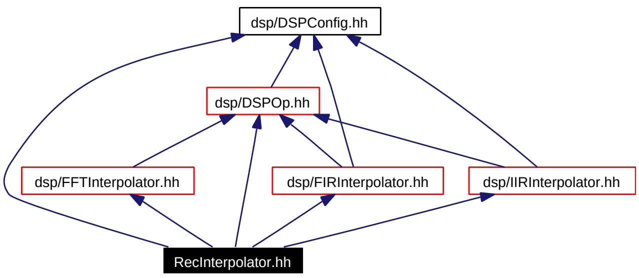
Include dependency graph for RecInterpolator.cc:



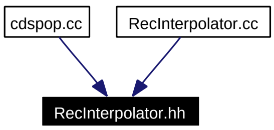
6.50 RecInterpolator.hh File Reference

```
#include "dsp/DSPConfig.hh"
#include "dsp/DSPOp.hh"
#include "dsp/FFTInterpolator.hh"
#include "dsp/FIRInterpolator.hh"
#include "dsp/IIRInterpolator.hh"
```

Include dependency graph for RecInterpolator.hh:



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



Classes

- class clRecInterpolator

  Recursive interpolation filter class implementation.

Defines

- #define RECINT_MAX_SUB_ROUNDS 32

  Maximum number of subsequent rounds.

6.50.1 Define Documentation

6.50.1.1 #define RECINT_MAX_SUB_ROUNDS 32

Maximum number of subsequent rounds.

Definition at line 33 of file RecInterpolator.hh.

Referenced by clRecInterpolator::Initialize().

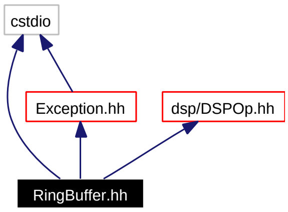
6.51 RingBuffer.hh File Reference

`#include <cstdio>`

`#include <Exception.hh>`

`#include "dsp/DSPOp.hh"`

Include dependency graph for RingBuffer.hh:



Classes

- class clXRingBuffer

  Exception class for clRingBuffer.

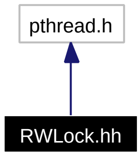
- class clRingBuffer

  Class for predefined size lockless ring buffer.

6.52 RWLock.hh File Reference

`#include <pthread.h>`

Include dependency graph for RWLock.hh:



Classes

- class clRWLock

  Class implementation of SUSv2 read-write locks.

Defines

- #define _XOPEN_SOURCE 600

6.52.1 Define Documentation

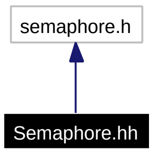
6.52.1.1 #define _XOPEN_SOURCE 600

Definition at line 29 of file RWLock.hh.

6.53 Semaphore.hh File Reference

```
#include <semaphore.h>
```

Include dependency graph for Semaphore.hh:



Classes

- class clSemaphore

    Class implementation of POSIX counting semaphores.

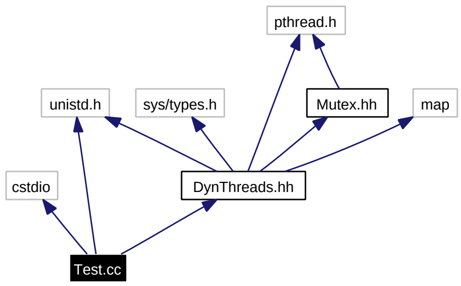
6.54 Test.cc File Reference

`#include <cstdio>`

`#include <unistd.h>`

`#include "DynThreads.hh"`

Include dependency graph for Test.cc:



Classes

- class clUserThreads

Functions

- int main (int argc, char ∗argv[ ])

6.54.1 Function Documentation

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

Definition at line 21 of file Test.cc.

References clDynThreads< TThreads >::Create(), clUserThreads::Stop(), and clDynThreadsBase::Wait().

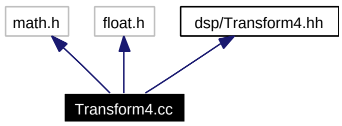
6.55 Transform4.cc File Reference

`#include <math.h>`

`#include <float.h>`

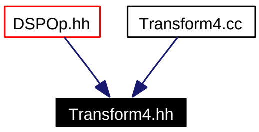
`#include "dsp/Transform4.hh"`

Include dependency graph for Transform4.cc:



6.56 Transform4.hh File Reference

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



Classes

- class clTransform4

  Decimation-in-frequency radix-2/4 transform.

Defines

- #define T4_INLINE

6.56.1 Define Documentation

6.56.1.1 #define T4_INLINE

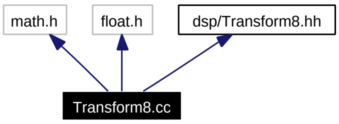
Definition at line 37 of file Transform4.hh.

Referenced by clTransform4::bitrv2(), clTransform4::bitrv2conj(), clTransform4::cft1st(), clTransform4::cftbsub(), clTransform4::cftfsub(), clTransform4::cftmdl(), clTransform4::dctsub(), clTransform4::dstsub(), clTransform4::rftbsub(), and clTransform4::rftfsub().

6.57 Transform8.cc File Reference

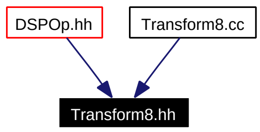
```
#include <math.h>
#include <float.h>
#include "dsp/Transform8.hh"
```

Include dependency graph for Transform8.cc:



6.58 Transform8.hh File Reference

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



Classes

- class clTransform8

    Decimation-in-frequency radix-2/4/8 transform.

Defines

- #define T8_INLINE

6.58.1 Define Documentation

6.58.1.1 #define T8_INLINE

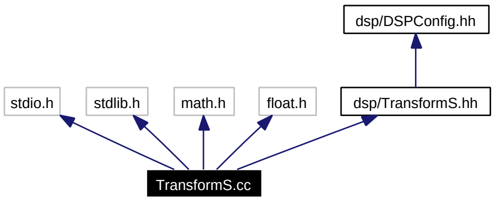
Definition at line 37 of file Transform8.hh.

Referenced by clTransform8::bitrv2(), clTransform8::bitrv2conj(), clTransform8::cft1st(), clTransform8::cftbsub(), clTransform8::cftfsub(), clTransform8::cftmdl(), clTransform8::dctsub(), clTransform8::dstsub(), clTransform8::rftbsub(), and clTransform8::rftfsub().

6.59 TransformS.cc File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <float.h>
#include "dsp/TransformS.hh"
```

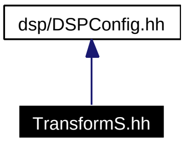
Include dependency graph for TransformS.cc:



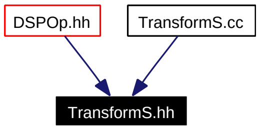
6.60 TransformS.hh File Reference

`#include "dsp/DSPConfig.hh"`

Include dependency graph for TransformS.hh:



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



Classes

- class clTransformS

  Decimation-in-frequency split-radix transform.

Defines

- #define TS_INLINE

6.60.1 Define Documentation

6.60.1.1 #define TS_INLINE

Definition at line 41 of file TransformS.hh.

Referenced by clTransformS::bitrv2(), clTransformS::bitrv208(), clTransformS::bitrv208neg(), clTransformS::bitrv216(), clTransformS::bitrv216neg(), clTransformS::bitrv2conj(), clTransformS::cftb040(), clTransformS::cftb1st(), clTransformS::cftf040(), clTransformS::cftf081(), clTransformS::cftf082(), clTransformS::cftf161(), clTransformS::cftf162(), clTransformS::cftf1st(), clTransformS::cftfx41(), clTransformS::cftleaf(), clTransformS::cftmdl1(), clTransformS::cftmdl2(), clTransformS::cftrec4(), clTransformS::cfttree(), clTransformS::cftx020(), clTransformS::dctsub(), clTransformS::dstsub(), clTransformS::rftbsub(), and clTransformS::rftfsub().

6.61 X86-64.c File Reference

6.62 X86-64.h File Reference

6.63 X86.c File Reference

6.64 X86.h File Reference

Index