

My Project

Generated by Doxygen 1.8.8

Sun Apr 10 2016 22:37:10

Contents

1	Bug List	1
2	Namespace Index	3
2.1	Namespace List	3
3	Hierarchical Index	5
3.1	Class Hierarchy	5
4	Class Index	7
4.1	Class List	7
5	File Index	9
5.1	File List	9
6	Namespace Documentation	11
6.1	radio Namespace Reference	11
6.1.1	Detailed Description	12
6.1.2	Enumeration Type Documentation	12
6.1.2.1	Age	12
6.1.2.2	Argument	12
6.1.2.3	Fractional	12
6.1.2.4	ModulationType	13
6.1.3	Function Documentation	13
6.1.3.1	aconj	13
6.1.3.2	fft	13
6.1.3.3	hilbert	14
6.1.3.4	ifft	15
6.1.3.5	makeIQ	16
6.1.3.6	ShowHelp	17
6.1.3.7	to_sint32	17
6.1.3.8	to_type	17
6.1.4	Variable Documentation	18
6.1.4.1	F_BASEBAND	18

6.1.4.2	F_LOWERSIDEBAND	18
6.1.4.3	F_UPPERSIDEBAND	19
6.1.4.4	FREQ_INTERMEDIATE	19
6.1.4.5	SAMPLING_RATE	19
7	Class Documentation	21
7.1	radio::Filter Class Reference	21
7.1.1	Detailed Description	21
7.1.2	Constructor & Destructor Documentation	21
7.1.2.1	Filter	21
7.1.3	Member Function Documentation	22
7.1.3.1	Pass	22
7.1.4	Member Data Documentation	22
7.1.4.1	data	22
7.1.4.2	diffEq	22
7.1.4.3	eqLength	22
7.1.4.4	size	22
7.2	radio::Gain Class Reference	23
7.2.1	Detailed Description	23
7.2.2	Constructor & Destructor Documentation	23
7.2.2.1	Gain	23
7.2.3	Member Function Documentation	23
7.2.3.1	Apply	23
7.3	radio::Modulator Class Reference	24
7.3.1	Detailed Description	24
7.3.2	Constructor & Destructor Documentation	24
7.3.2.1	Modulator	24
7.3.2.2	~Modulator	24
7.3.3	Member Function Documentation	24
7.3.3.1	Mod	24
7.4	radio::PITone Class Reference	25
7.4.1	Detailed Description	26
7.4.2	Constructor & Destructor Documentation	26
7.4.2.1	PITone	26
7.4.3	Member Function Documentation	26
7.4.3.1	Add	26
7.5	radio::Sinusoid Class Reference	27
7.5.1	Detailed Description	28
7.5.2	Constructor & Destructor Documentation	28
7.5.2.1	Sinusoid	28

7.5.2.2	~Sinusoid	28
7.5.3	Member Function Documentation	28
7.5.3.1	next	28
7.5.3.2	nextShifted	29
7.5.4	Member Data Documentation	29
7.5.4.1	frequency	29
7.5.4.2	samplingRate	29
7.5.4.3	sinIndex	29
7.5.4.4	sinIndexShifted	29
7.5.4.5	sinusoid	29
7.5.4.6	sinusoidShift90	29
8	File Documentation	31
8.1	etc/doxygen.config File Reference	31
8.2	makefile File Reference	31
8.3	src/alsa_test.cpp File Reference	31
8.3.1	Detailed Description	31
8.3.2	Function Documentation	32
8.3.2.1	main	32
8.4	src/auxiliary.hpp File Reference	32
8.5	src/baseband_filter_test.cpp File Reference	33
8.5.1	Function Documentation	33
8.5.1.1	main	33
8.6	src/definitions.hpp File Reference	34
8.6.1	Macro Definition Documentation	35
8.6.1.1	ENUM	35
8.6.1.2	ERROR	35
8.6.2	Typedef Documentation	35
8.6.2.1	byte	35
8.6.2.2	cfloat32	36
8.6.2.3	float32	36
8.6.2.4	float64	36
8.6.2.5	fparams	36
8.6.2.6	sint16	36
8.6.2.7	sint32	36
8.6.2.8	sint64	36
8.6.2.9	sint8	36
8.6.2.10	uint16	36
8.6.2.11	uint32	36
8.6.2.12	uint64	36

8.6.2.13	uint8	36
8.7	src/fft_test.cpp File Reference	37
8.7.1	Detailed Description	37
8.7.2	Typedef Documentation	37
8.7.2.1	CArray	37
8.7.3	Function Documentation	38
8.7.3.1	fft	38
8.7.3.2	hilbert	38
8.7.3.3	ifft	38
8.7.3.4	main	39
8.7.4	Variable Documentation	39
8.7.4.1	PI	39
8.8	src/fft_test2.cpp File Reference	39
8.8.1	Detailed Description	40
8.8.2	Function Documentation	40
8.8.2.1	main	40
8.9	src/Filter.hpp File Reference	41
8.9.1	Detailed Description	42
8.10	src/fvectors.hpp File Reference	42
8.10.1	Detailed Description	43
8.11	src/Gain.hpp File Reference	44
8.12	src/iq_test.cpp File Reference	45
8.12.1	Detailed Description	45
8.12.2	Function Documentation	45
8.12.2.1	main	45
8.13	src/lsb_filter_test.cpp File Reference	46
8.13.1	Detailed Description	46
8.13.2	Function Documentation	47
8.13.2.1	main	47
8.14	src/main.cpp File Reference	47
8.14.1	Detailed Description	47
8.14.2	Function Documentation	48
8.14.2.1	main	48
8.15	src/mic_test.cpp File Reference	48
8.15.1	Detailed Description	49
8.15.2	Function Documentation	49
8.15.2.1	main	49
8.16	src/Modulator.hpp File Reference	49
8.17	src/modulator_test.cpp File Reference	50
8.17.1	Function Documentation	51

8.17.1.1	main	51
8.18	src/multi_sinusoid_test.cpp File Reference	51
8.18.1	Function Documentation	52
8.18.1.1	main	52
8.19	src/piped_test.cpp File Reference	52
8.19.1	Detailed Description	53
8.19.2	Function Documentation	53
8.19.2.1	main	53
8.20	src/pl_tone_test.cpp File Reference	53
8.20.1	Function Documentation	54
8.20.1.1	main	54
8.21	src/PITone.hpp File Reference	54
8.21.1	Detailed Description	55
8.22	src/Sinusoid.hpp File Reference	55
8.22.1	Detailed Description	56
8.23	src/sinusoid_test.cpp File Reference	57
8.23.1	Function Documentation	57
8.23.1.1	main	57
8.24	src/zdomain.hpp File Reference	57
8.24.1	Detailed Description	58
Index		59

Chapter 1

Bug List

File [alsa_test.cpp](#)

clicking noise from sinusoidal discontinuity

Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

radio	Helper-functions for main()	11
-----------------------	---	--------------------

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

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

radio::Filter	21
radio::Gain	23
radio::Modulator	24
radio::Sinusoid	27
radio::PITone	25

Chapter 4

Class Index

4.1 Class List

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

radio::Filter	21
radio::Gain	23
radio::Modulator	24
radio::PITone	25
radio::Sinusoid	27

Chapter 5

File Index

5.1 File List

Here is a list of all files with brief descriptions:

makefile	31
etc/doxygen.config	31
src/alsa_test.cpp	
Tests sinusoidal tone generation	31
src/auxiliary.hpp	32
src/baseband_filter_test.cpp	33
src/definitions.hpp	34
src/fft_test.cpp	
Tests FFT, IFFT, and Hilbert implementations	37
src/fft_test2.cpp	
Tests FFT, IFFT, and Hilbert implementations in zdomain.hpp	39
src/Filter.hpp	
Defines the Filter class	41
src/fvectors.hpp	
Defines the transfer function coefficients used in the instances of the Filter class in this program	42
src/Gain.hpp	44
src/iq_test.cpp	
Generates test IQ signal	45
src/lsb_filter_test.cpp	
Program to test the LSB-via-filter implementation	46
src/main.cpp	
"brains" of the entire project	47
src/mic_test.cpp	
Tests getting mic input via ALSA May not even compile at the moment	48
src/Modulator.hpp	49
src/modulator_test.cpp	50
src/multi_sinusoid_test.cpp	51
src/piped_test.cpp	
Contains the original program used to test the piping-in idea	52
src/pl_tone_test.cpp	53
src/PITone.hpp	
PITone class	54
src/Sinusoid.hpp	
Sinusoid class	55
src/sinusoid_test.cpp	57
src/zdomain.hpp	
Contains the functions to manipulate sequential data in the frequency (z) domain	57

Chapter 6

Namespace Documentation

6.1 radio Namespace Reference

contains helper-functions for [main\(\)](#)

Classes

- class [Filter](#)
- class [Gain](#)
- class [Modulator](#)
- class [PITone](#)
- class [Sinusoid](#)

Enumerations

- enum [Age](#) { [OLD](#), [NEW](#) }
- enum [Fractional](#) { [NUM](#), [DEN](#) }
- enum [Argument](#) { [FREQ](#) = 1, [MODE](#), [PL_TONE](#) }
- enum [ModulationType](#) {
 [ModulationType::DSB_LC](#), [ModulationType::DSB_SC](#), [ModulationType::USB_FILTERED](#), [ModulationType::USB_HILBERT](#),
 [ModulationType::LSB_FILTERED](#), [ModulationType::LSB_HILBERT](#), [ModulationType::FM_NARROW](#),
 [ModulationType::FM_WIDE](#) }

Functions

- void [ShowHelp](#) ()
- void [to_sint32](#) ([float32](#) *data, [uint32](#) size)
- [ModulationType](#) [to_type](#) (std::string str)
- void [aconj](#) ([cfloat32](#) *data, [uint32](#) size)
- void [fft](#) ([cfloat32](#) *data, [uint32](#) size)
- void [hilbert](#) ([float32](#) *data, [float32](#) *dest, [uint32](#) size)
- void [ifft](#) ([cfloat32](#) *data, [uint32](#) size)
- void [makeIQ](#) ([float32](#) *data, [float32](#) *dest, [uint32](#) size)

Variables

- [fparams F_BASEBAND](#)
- [fparams F_LOWERSIDEBAND](#)
- [fparams F_UPPERSIDEBAND](#)
- const [uint32](#) [FREQ_INTERMEDIATE](#) = 20000
- const [uint32](#) [SAMPLING_RATE](#) = 48000

6.1.1 Detailed Description

contains helper-functions for [main\(\)](#)

Contains the classes for the various types of modulation supported by the program.

Author

Samuel Andrew Wisner, awisner94@gmail.com

This namespace contains all the classes, functions, and enumerations used in the application.

6.1.2 Enumeration Type Documentation

6.1.2.1 enum [radio::Age](#)

Describes the age of a filter (from last [Pass\(\)](#) or in this [Pass\(\)](#))

Enumerator

OLD

NEW

Definition at line 51 of file [definitions.hpp](#).

6.1.2.2 enum [radio::Argument](#)

Enumerator

FREQ

MODE

PL_TONE

Definition at line 61 of file [definitions.hpp](#).

6.1.2.3 enum [radio::Fractional](#)

Describes the numerator and denominator of a z-domain transfer function

Enumerator

NUM

DEN

Definition at line 56 of file [definitions.hpp](#).

6.1.2.4 enum `radio::ModulationType` `[strong]`

Describes a form of modulation.

Enumerator

DSB_LC
DSB_SC
USB_FILTERED
USB_HILBERT
LSB_FILTERED
LSB_HILBERT
FM_NARROW
FM_WIDE

Definition at line 66 of file definitions.hpp.

6.1.3 Function Documentation

6.1.3.1 void `radio::aconj` (`cfloat32 * data`, `uint32 size`)

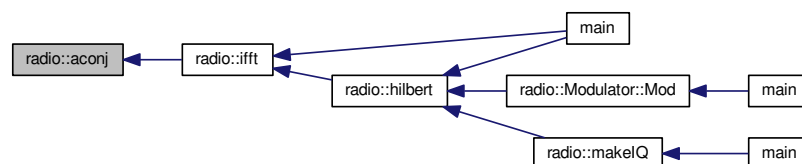
Replaces the values in an array of complex float32's with their respective conjugates.

Parameters

<i>data</i>	the array whose values should be replaced with their respective conjugates
<i>size</i>	the number of elements in the data array

Definition at line 84 of file zdomain.hpp.

Here is the caller graph for this function:

6.1.3.2 void `radio::fft` (`cfloat32 * data`, `uint32 size`)

Replaces the values of an array of `cfloat32`'s with the array's DFT using a decimation-in-frequency algorithm.

This code is based on code from http://rosettacode.org/wiki/Fast_Fourier_transform#C.↵2B.2B.

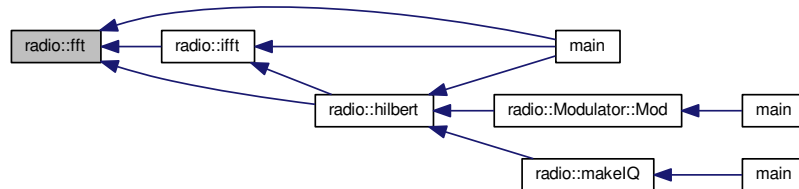
Parameters

<i>data</i>	the array whose values should be replaced with its DFT
-------------	--

<i>size</i>	the number of elements in the data array
-------------	--

Definition at line 90 of file zdomain.hpp.

Here is the caller graph for this function:



6.1.3.3 void radio::hilbert (float32 * data, float32 * dest, uint32 size)

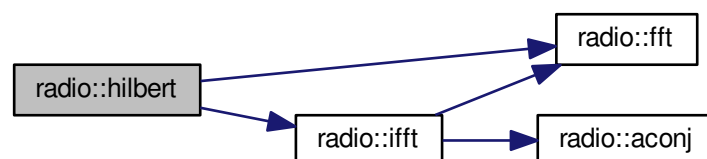
Performs the hilbert tranfor of an array of float32's.

Parameters

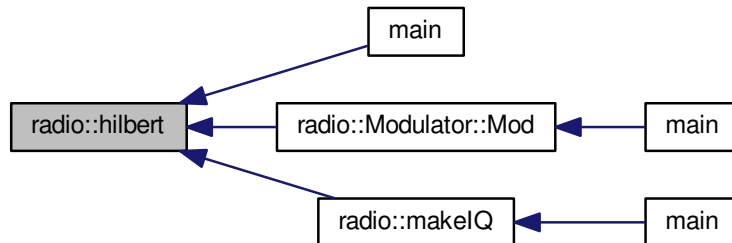
<i>data</i>	the source array of the REAL numbers of which to take the Hilbert transform
<i>dest</i>	the destination array of REAL numbers for the results of the Hilbert transform
<i>size</i>	the number of elements in the data and dest arrays

Definition at line 138 of file zdomain.hpp.

Here is the call graph for this function:



Here is the caller graph for this function:



6.1.3.4 void radio::ifft (cfloat32 * data, uint32 size)

Replaces the values of an array of `cfloat32`'s with the array's inverse DFT.

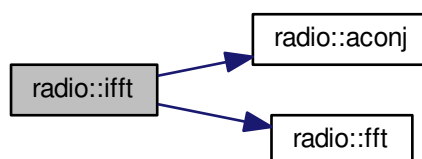
This code is based on code from http://rosettacode.org/wiki/Fast_Fourier_transform#C.↔2B.2B.

Parameters

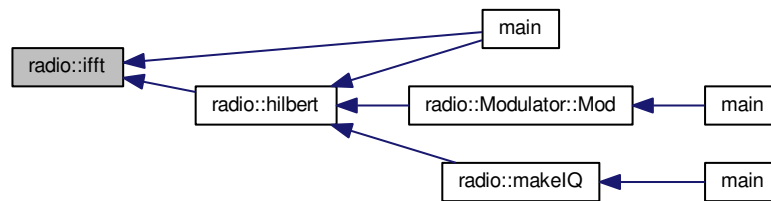
<i>data</i>	the array whose values should be replaced with its inverse DFT
<i>size</i>	the number of elements in the data array

Definition at line 161 of file `zdomain.hpp`.

Here is the call graph for this function:



Here is the caller graph for this function:



6.1.3.5 void radio::makeIQ (float32 * data, float32 * dest, uint32 size)

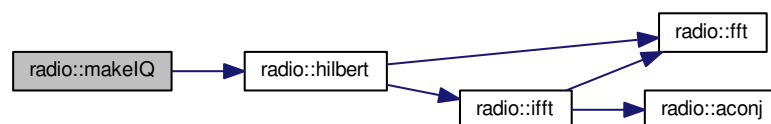
Produces an interleaved array of first an element from an original array of data and then an element from the original data's Hilbert transform. This function is intended to generate a two-channel output (I/Q output) for mixing applications.

Parameters

<i>data</i>	the original data (left channel)
<i>dest</i>	the interleaved data (left channel original data, right channel transformed data) twice the size of the original data array
<i>size</i>	the number of elements in the data array (NOT in the destination array)

Definition at line 171 of file `zdomain.hpp`.

Here is the call graph for this function:



Here is the caller graph for this function:



6.1.3.6 void radio::ShowHelp ()

Displays the help information.

Definition at line 21 of file auxiliary.hpp.

Here is the caller graph for this function:



6.1.3.7 void radio::to_sint32 (float32 * data, uint32 size)

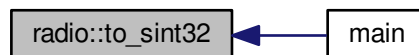
Converts float32 samples to sint32 samples. Rounds conversion to nearest integer.

Parameters

<i>data</i>	the array containing the float32 samples that are directly replaced by their respective sint32 representations
<i>size</i>	the number of elements in the data array

Definition at line 61 of file auxiliary.hpp.

Here is the caller graph for this function:



6.1.3.8 ModulationType radio::to_type (std::string str)

Converts a string representation of the supported modulation types (see [ShowHelp\(\)](#) documentation) to the enum ModulationType value.

This function is not as elegant as it could be. Ideally, I would have used a `std::map<string, ModulationType>` rather than a long series of if-else's.

Parameters

<i>str</i>	type of modulation in typed form
------------	----------------------------------

Returns

enum value of the type of modulation

Definition at line 79 of file auxiliary.hpp.

Here is the caller graph for this function:

**6.1.4 Variable Documentation****6.1.4.1 fparams radio::F_BASEBAND****Initial value:**

```

= { std::vector<float64> {
    0.0008977019461,
    -0.002215694636,
    0.001372192986,
    0.001372192986,
    -0.002215694636,
    0.0008977019461
  }, std::vector<float64> {
    1,
    -4.678616047,
    8.822912216,
    -8.379911423,
    4.007629871,
    -0.7719064355
  } }
  
```

Baseband filter coefficients. Generated with MATLAB 2015A.

Definition at line 19 of file fvectors.hpp.

6.1.4.2 fparams radio::F_LOWERSIDEBAND**Initial value:**

```

= { std::vector<float64> {
    0.2758039069174,
    2.763578787693,
    12.83915022756,
    36.47584850651,
    70.37084637368,
    96.76893503179,
    96.76893503179,
    70.37084637368,
    36.47584850651,
    12.83915022756,
    2.763578787693,
    0.2758039069174
  }, std::vector<float64> {
    1,
    7.605497780083,
    27.34180552438,
    60.83375457605,
    92.60908886875,
    100.8363857,
    79.74796574736,
  } }
  
```

```

        45.4982252145,
        18.13566776308,
        4.690036472717,
        0.6617552879305,
        0.0281427334611
    } }

```

Lower-sideband filter coefficients. Generated with MATLAB 2015A.

Definition at line 38 of file fvectors.hpp.

6.1.4.3 fparams radio::F_UPPERSIDEBAND

Initial value:

```

= { std::vector<float64> {
    0.001690387681463,
    0.01145271586989,
    0.03591799189724,
    0.06576926098562,
    0.07119343282702,
    0.03156377419766,
    -0.03156377419766,
    -0.07119343282702,
    -0.06576926098562,
    -0.03591799189724,
    -0.01145271586989,
    -0.001690387681463
}, std::vector<float64> {
    1,
    9.465175013624,
    41.62402815905,
    112.0971027069,
    205.2097686473,
    267.9378582311,
    254.486805213,
    175.7772755115,
    86.51619894548,
    28.89988093561,
    5.89781461091,
    0.5572910543053
} }

```

Upper-sideband filter coefficients. Generated with MATLAB 2015A.

Definition at line 69 of file fvectors.hpp.

6.1.4.4 const uint32 radio::FREQ_INTERMEDIATE = 20000

The default intermediate carrier frequency

Definition at line 26 of file Modulator.hpp.

6.1.4.5 const uint32 radio::SAMPLING_RATE = 48000

The default sampling rate (frequency)

Definition at line 31 of file Modulator.hpp.

Chapter 7

Class Documentation

7.1 radio::Filter Class Reference

```
#include <Filter.hpp>
```

Public Member Functions

- [Filter](#) ([float32](#) *[data](#), [uint32](#) [size](#), [fparams](#) &[diffEq](#))
- void [Pass](#) ()

Protected Attributes

- [uint8](#) [eqLength](#)
- [uint32](#) [size](#)
- [float32](#) * [data](#)
- [fparams](#) [diffEq](#)

7.1.1 Detailed Description

This class implements a z-domain filter on a specified array of float32's (a.k.a. singles, floats). It requires the transfer function coefficients already be calculated (i.e., it does not generate the coefficients based on desired filter characteristics). MATLAB and its Signal Processing Toolbox can be used to generate the coefficients.

While this class is designed to implement a single-section filter, several instances of the class can be created and run over the data array sequentially to effectively implement a multi-section filter.

Definition at line 27 of file Filter.hpp.

7.1.2 Constructor & Destructor Documentation

7.1.2.1 radio::Filter::Filter ([float32](#) * [data](#), [uint32](#) [size](#), [fparams](#) & [diffEq](#))

Initializes [Filter](#) based on a difference equation.

Parameters

<i>data</i>	array to be filtered. The filtered data will be placed here.
<i>size</i>	number of elements in the data array
<i>diffEq</i>	a vector containing two vectors of float32's (a.k.a. singles, floats), containing the numerator and denominator coefficients, respectively, of the z-domain transfer function of the filter in descending order (z^0 , z^{-1} , z^{-2} , etc.).

Definition at line 79 of file Filter.hpp.

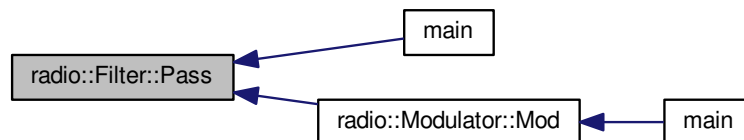
7.1.3 Member Function Documentation

7.1.3.1 void radio::Filter::Pass ()

Passes the data array through the digital filter but does not account for previous $x[n]$ and $y[n]$ values from the previous call to [Pass\(\)](#).

Definition at line 86 of file Filter.hpp.

Here is the caller graph for this function:



7.1.4 Member Data Documentation

7.1.4.1 float32* radio::Filter::data [protected]

A pointer to the data array that should be filtered when [Pass\(\)](#) is called.

Definition at line 68 of file Filter.hpp.

7.1.4.2 fparams radio::Filter::diffEq [protected]

A vector containing two vectors of float32's (a.k.a. singles, floats), containing the numerator and denominator coefficients, respectively, of the z-domain transfer function of the filter in descending order (z^0 , z^{-1} , z^{-2} , etc.).

Definition at line 76 of file Filter.hpp.

7.1.4.3 uint8 radio::Filter::eqLength [protected]

The number of terms in the numerator (or denominator) of the transfer function.

Definition at line 57 of file Filter.hpp.

7.1.4.4 uint32 radio::Filter::size [protected]

The number of elements in the data array.

Definition at line 62 of file Filter.hpp.

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

- [src/Filter.hpp](#)

7.2 radio::Gain Class Reference

```
#include <Gain.hpp>
```

Public Member Functions

- [Gain](#) ([float32](#) *data, [uint32](#) size, [float32](#) gaindB)
- void [Apply](#) ()

7.2.1 Detailed Description

Applies a gain to a (baseband) signal.

Definition at line 12 of file Gain.hpp.

7.2.2 Constructor & Destructor Documentation

7.2.2.1 radio::Gain::Gain ([float32](#) * data, [uint32](#) size, [float32](#) gaindB)

Initializes a [Gain](#) object and converts gain from decibels to a standard value.

Parameters

<i>data</i>	the signal to which the gain is applied
<i>size</i>	the number of elements in the data array
<i>gaindB</i>	the desired gain in decibels (of power)

Definition at line 55 of file Gain.hpp.

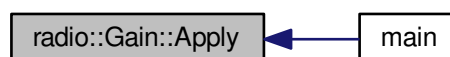
7.2.3 Member Function Documentation

7.2.3.1 void radio::Gain::Apply ()

Applies the gain to the signal contained in the data array

Definition at line 61 of file Gain.hpp.

Here is the caller graph for this function:



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

- [src/Gain.hpp](#)

7.3 radio::Modulator Class Reference

```
#include <Modulator.hpp>
```

Public Member Functions

- [Modulator](#) (float32 data[], uint32 size, [ModulationType](#) type, float32 freqInter=FREQ_INTERMEDIATE, uint32 rate=SAMPLING_RATE)
- [~Modulator](#) ()
- void [Mod](#) ()

7.3.1 Detailed Description

This class, while not intended to be called directly, is a superclass for the classes of the modulation forms used in this project.

Definition at line 37 of file Modulator.hpp.

7.3.2 Constructor & Destructor Documentation

7.3.2.1 `radio::Modulator::Modulator (float32 data[], uint32 size, ModulationType type, float32 freqInter = FREQ_INTERMEDIATE, uint32 rate = SAMPLING_RATE)`

Creates a [Modulator](#) with the specified parameters. Intended to be called only by subclasses.

Parameters

<i>freqInter</i>	the frequency of the IF carrier sinusoid
<i>rate</i>	the sampling rate of the baseband and IF signals
<i>data</i>	the array holding initially the baseband signal
<i>size</i>	the number of elements in data
<i>type</i>	form of modulation to use

Definition at line 101 of file Modulator.hpp.

7.3.2.2 `radio::Modulator::~~Modulator ()`

Definition at line 115 of file Modulator.hpp.

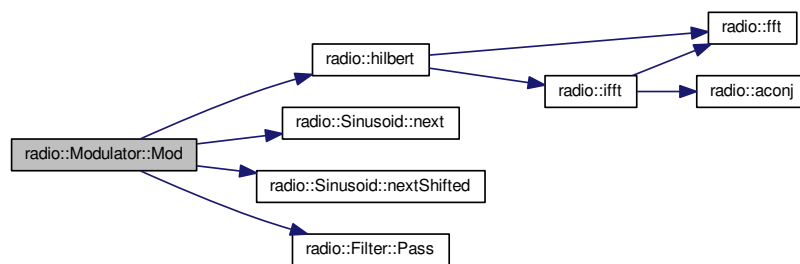
7.3.3 Member Function Documentation

7.3.3.1 `void radio::Modulator::Mod ()`

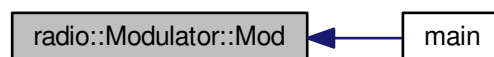
Modulates the audio currently in the data array.

Definition at line 119 of file Modulator.hpp.

Here is the call graph for this function:



Here is the caller graph for this function:



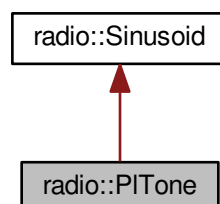
The documentation for this class was generated from the following file:

- [src/Modulator.hpp](#)

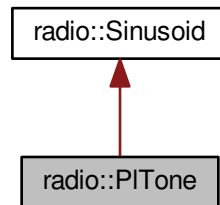
7.4 radio::PITone Class Reference

```
#include <PlTone.hpp>
```

Inheritance diagram for `radio::PITone`:



Collaboration diagram for radio::PITone:



Public Member Functions

- `PITone` (`float32` amplitude, `float32 *`data, `uint32` size, `float32` frequency, `uint32` samplingRate)
- void `Add` ()

7.4.1 Detailed Description

This class creates a CTCSS subcarrier (PL tone) at a specified frequency in a baseband signal.

Definition at line 18 of file `PITone.hpp`.

7.4.2 Constructor & Destructor Documentation

7.4.2.1 `radio::PITone::PITone (float32 amplitude, float32 * data, uint32 size, float32 frequency, uint32 samplingRate)`

Creates a `PITone` object.

Parameters

<i>amplitude</i>	the amplitude (0-1) of the subcarrier. Assumes baseband signal has a peak-to-peak range of -1 to 1.
<i>data</i>	an array containing a portion of the discrete baseband signal
<i>size</i>	the number of elements in the data array
<i>frequency</i>	the frequency of the CTCSS tone in the baseband (not in the IF or RF signals)
<i>samplingRate</i>	the sampling frequency of the baseband signal

Definition at line 63 of file `PITone.hpp`.

7.4.3 Member Function Documentation

7.4.3.1 `void radio::PITone::Add ()`

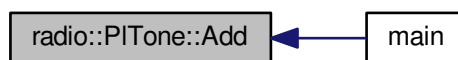
Adds the CTCSS tone to the baseband signal.

Definition at line 75 of file `PITone.hpp`.

Here is the call graph for this function:



Here is the caller graph for this function:



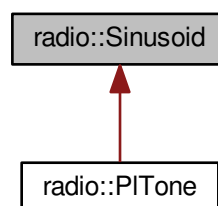
The documentation for this class was generated from the following file:

- [src/PITone.hpp](#)

7.5 radio::Sinusoid Class Reference

```
#include <Sinusoid.hpp>
```

Inheritance diagram for `radio::Sinusoid`:



Public Member Functions

- [Sinusoid](#) ([float32 frequency](#), [uint32 samplingRate=48000](#))
- [~Sinusoid](#) ()
- [float32 next](#) ()
- [float32 nextShifted](#) ()

Protected Attributes

- [float32 frequency](#)
- [uint32 sinIndex](#) = 0
- [uint32 sinIndexShifted](#) = 0
- [uint32 samplingRate](#)
- [float32 * sinusoid](#)
- [float32 * sinusoidShift90](#)

7.5.1 Detailed Description

This class creates an easy-to-call sinusoid that will preserve its phase throughout its lifespan. Essentially, it is a ring buffer.

Definition at line 20 of file Sinusoid.hpp.

7.5.2 Constructor & Destructor Documentation

7.5.2.1 `radio::Sinusoid::Sinusoid (float32 frequency, uint32 samplingRate = 48000)`

Creates a ring-buffer sinusoid.

Definition at line 77 of file Sinusoid.hpp.

7.5.2.2 `radio::Sinusoid::~Sinusoid ()`

Free arrays malloc'd in the constructor.

Definition at line 92 of file Sinusoid.hpp.

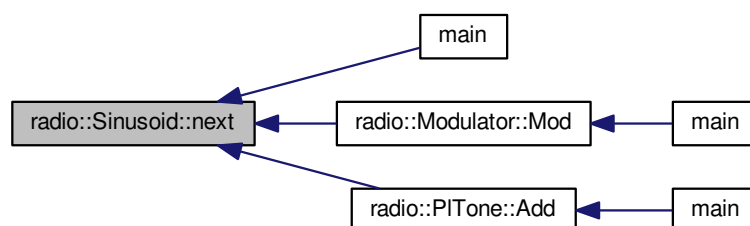
7.5.3 Member Function Documentation

7.5.3.1 `float32 radio::Sinusoid::next ()`

Provides the next value of the sinusoid in a manner consistent with a ring buffer.

Definition at line 97 of file Sinusoid.hpp.

Here is the caller graph for this function:



7.5.3.2 float32 radio::Sinusoid::nextShifted ()

Provides the next value of the sinusoid shifted 90 degrees in a manner consistant with a ring buffer.

Definition at line 102 of file Sinusoid.hpp.

Here is the caller graph for this function:



7.5.4 Member Data Documentation

7.5.4.1 float32 radio::Sinusoid::frequency [protected]

The frequency of the sinusoid

Definition at line 48 of file Sinusoid.hpp.

7.5.4.2 uint32 radio::Sinusoid::samplingRate [protected]

The sampling rate

Definition at line 63 of file Sinusoid.hpp.

7.5.4.3 uint32 radio::Sinusoid::sinIndex = 0 [protected]

The current index of the sinusoid's unshifted array

Definition at line 53 of file Sinusoid.hpp.

7.5.4.4 uint32 radio::Sinusoid::sinIndexShifted = 0 [protected]

The current index of the shifted sinusoid's array

Definition at line 58 of file Sinusoid.hpp.

7.5.4.5 float32* radio::Sinusoid::sinusoid [protected]

Initialized as an array of the sinusoid values

Definition at line 68 of file Sinusoid.hpp.

7.5.4.6 float32* radio::Sinusoid::sinusoidShift90 [protected]

Initialized as an array of the sinusoid values shifted 90 degrees

Definition at line 74 of file Sinusoid.hpp.

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

- [src/Sinusoid.hpp](#)

Chapter 8

File Documentation

8.1 etc/doxygen.config File Reference

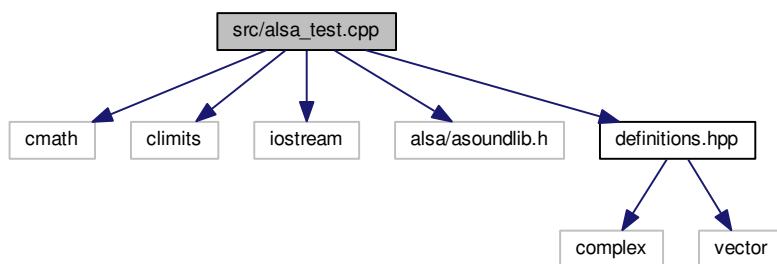
8.2 makefile File Reference

8.3 src/alsa_test.cpp File Reference

Tests sinusoidal tone generation.

```
#include <cmath>
#include <climits>
#include <iostream>
#include <alsa/asoundlib.h>
#include "definitions.hpp"
```

Include dependency graph for alsa_test.cpp:



Functions

- int `main` ()

8.3.1 Detailed Description

Tests sinusoidal tone generation.

Author

Samuel Andrew Wisner, awisner94@gmail.com

Bug clicking noise from sinusoidal discontinuity

Definition in file [alsa_test.cpp](#).

8.3.2 Function Documentation**8.3.2.1 int main ()**

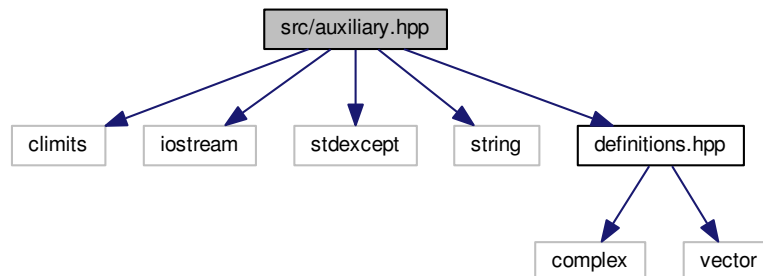
This program tests sinusoidal speaker output through the ALSA API. Not sure if it works. When it did at least compile and run, it produced a sinusoid with an approximately twice-per-second clicking noise.

Definition at line 22 of file [alsa_test.cpp](#).

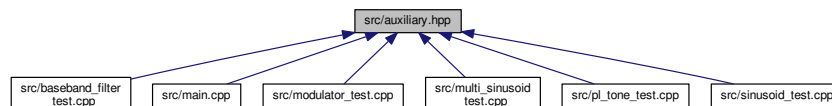
8.4 src/auxiliary.hpp File Reference

```
#include <climits>
#include <iostream>
#include <stdexcept>
#include <string>
#include "definitions.hpp"
```

Include dependency graph for auxiliary.hpp:



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

**Namespaces**

- [radio](#)
contains helper-functions for [main\(\)](#)

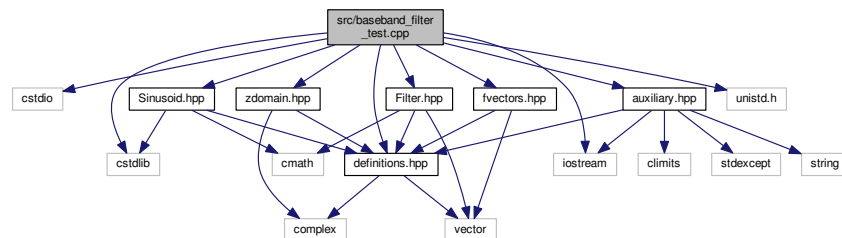
Functions

- void `radio::ShowHelp` ()
- void `radio::to_sint32` (float32 *data, uint32 size)
- ModulationType `radio::to_type` (std::string str)

8.5 src/baseband_filter_test.cpp File Reference

```
#include <cstdio>
#include <cstdlib>
#include <iostream>
#include <unistd.h>
#include "auxiliary.hpp"
#include "definitions.hpp"
#include "Filter.hpp"
#include "fvectors.hpp"
#include "Sinusoid.hpp"
#include "zdomain.hpp"
```

Include dependency graph for `baseband_filter_test.cpp`:



Functions

- int `main` (int argc, char *argv[])

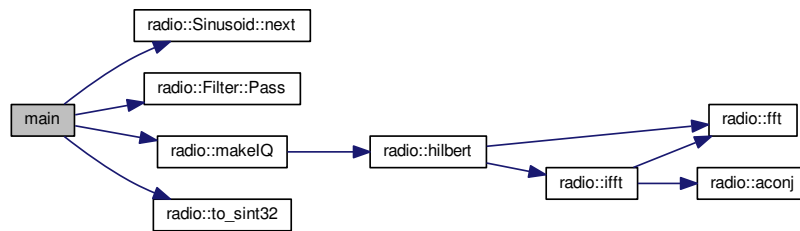
8.5.1 Function Documentation

8.5.1.1 int main (int argc, char * argv[])

Program to test the Filter class and the baseband filter coefficients.

Definition at line 24 of file `baseband_filter_test.cpp`.

Here is the call graph for this function:

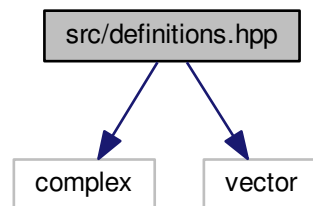


8.6 src/definitions.hpp File Reference

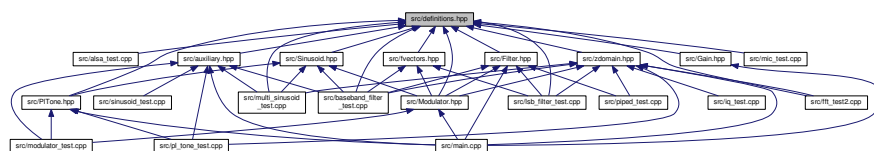
```
#include <complex>
```

```
#include <vector>
```

Include dependency graph for definitions.hpp:



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



Namespaces

- [radio](#)
contains helper-functions for [main\(\)](#)

Macros

- #define [ENUM](#) signed char

Contains declarations of system-independant (universal size) integers and float types, shortened type names for some commonly used types, and enumerations.

- `#define ERROR -1`

Typedefs

- `typedef unsigned char byte`
- `typedef unsigned char uint8`
- `typedef signed char sint8`
- `typedef unsigned short uint16`
- `typedef signed short sint16`
- `typedef unsigned int uint32`
- `typedef signed int sint32`
- `typedef unsigned long long uint64`
- `typedef signed long long sint64`
- `typedef float float32`
- `typedef double float64`
- `typedef std::complex< float32 > cfloat32`
- `typedef std::vector`
`< std::vector< float64 > > fparams`

Enumerations

- `enum radio::Age { radio::OLD, radio::NEW }`
- `enum radio::Fractional { radio::NUM, radio::DEN }`
- `enum radio::Argument { radio::FREQ = 1, radio::MODE, radio::PL_TONE }`
- `enum radio::ModulationType {`
`radio::ModulationType::DSB_LC, radio::ModulationType::DSB_SC, radio::ModulationType::USB_FILTERED,`
`radio::ModulationType::USB_HILBERT,`
`radio::ModulationType::LSB_FILTERED, radio::ModulationType::LSB_HILBERT, radio::ModulationType::FM_↵`
`M_NARROW, radio::ModulationType::FM_WIDE }`

8.6.1 Macro Definition Documentation

8.6.1.1 `#define ENUM signed char`

Contains declarations of system-independant (universal size) integers and float types, shortened type names for some commonly used types, and enumerations.

Author

Samuel Andrew Wisner, awisner94@gmail.com

Definition at line 14 of file definitions.hpp.

8.6.1.2 `#define ERROR -1`

Definition at line 15 of file definitions.hpp.

8.6.2 Typedef Documentation

8.6.2.1 `typedef unsigned char byte`

Definition at line 17 of file definitions.hpp.

8.6.2.2 `typedef std::complex<float32> cfloat32`

Defines a type for complex float32's.

Definition at line 36 of file definitions.hpp.

8.6.2.3 `typedef float float32`

Definition at line 30 of file definitions.hpp.

8.6.2.4 `typedef double float64`

Definition at line 31 of file definitions.hpp.

8.6.2.5 `typedef std::vector<std::vector<float64> > fparams`

Defines a type for the filter coefficients.

Definition at line 41 of file definitions.hpp.

8.6.2.6 `typedef signed short sint16`

Definition at line 22 of file definitions.hpp.

8.6.2.7 `typedef signed int sint32`

Definition at line 25 of file definitions.hpp.

8.6.2.8 `typedef signed long long sint64`

Definition at line 28 of file definitions.hpp.

8.6.2.9 `typedef signed char sint8`

Definition at line 19 of file definitions.hpp.

8.6.2.10 `typedef unsigned short uint16`

Definition at line 21 of file definitions.hpp.

8.6.2.11 `typedef unsigned int uint32`

Definition at line 24 of file definitions.hpp.

8.6.2.12 `typedef unsigned long long uint64`

Definition at line 27 of file definitions.hpp.

8.6.2.13 `typedef unsigned char uint8`

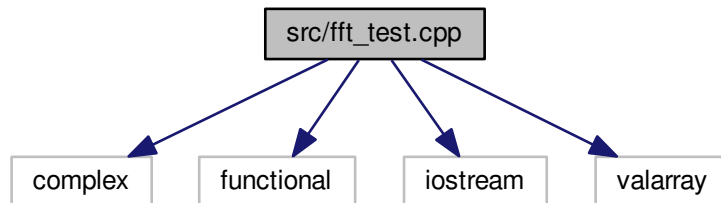
Definition at line 18 of file definitions.hpp.

8.7 src/fft_test.cpp File Reference

Tests FFT, IFFT, and Hilbert implementations.

```
#include <complex>
#include <functional>
#include <iostream>
#include <valarray>
```

Include dependency graph for `fft_test.cpp`:



Typedefs

- `typedef std::valarray< std::complex< double > > CArray`

Functions

- `void fft (CArray &x)`
- `void ifft (CArray &x)`
- `std::complex< double > hilbert (std::complex< double > n)`
- `int main ()`

Variables

- `const double PI = 3.141592653589793238460`

8.7.1 Detailed Description

Tests FFT, IFFT, and Hilbert implementations.

Author

Samuel Andrew Wisner, awisner94@gmail.com

Definition in file `fft_test.cpp`.

8.7.2 Typedef Documentation

8.7.2.1 `typedef std::valarray<std::complex<double> > CArray`

Definition at line 14 of file `fft_test.cpp`.

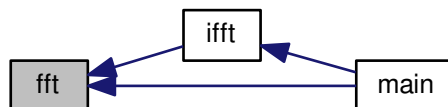
8.7.3 Function Documentation

8.7.3.1 void fft (CArray & x)

This code was taken from http://rosettacode.org/wiki/Fast_Fourier_transform#C.2B.2B.

Definition at line 23 of file fft_test.cpp.

Here is the caller graph for this function:



8.7.3.2 std::complex<double> hilbert (std::complex< double > n)

Definition at line 87 of file fft_test.cpp.

Here is the caller graph for this function:



8.7.3.3 void ifft (CArray & x)

Definition at line 72 of file fft_test.cpp.

Here is the call graph for this function:



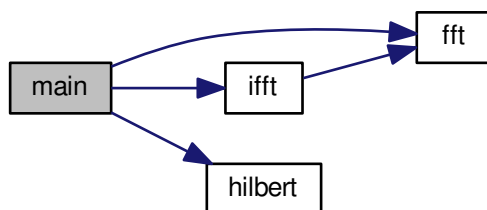
Here is the caller graph for this function:



8.7.3.4 int main ()

Definition at line 91 of file `fft_test.cpp`.

Here is the call graph for this function:



8.7.4 Variable Documentation

8.7.4.1 const double PI = 3.141592653589793238460

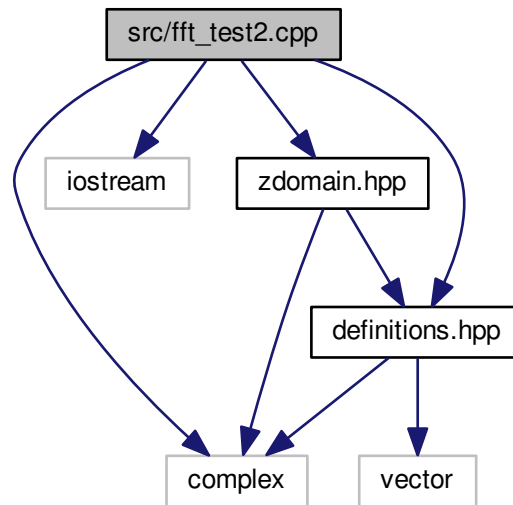
Definition at line 12 of file `fft_test.cpp`.

8.8 src/fft_test2.cpp File Reference

Tests FFT, IFFT, and Hilbert implementations in [zdomain.hpp](#).

```
#include <complex>
#include <iostream>
#include "definitions.hpp"
#include "zdomain.hpp"
```

Include dependency graph for `fft_test2.cpp`:



Functions

- `int main ()`

8.8.1 Detailed Description

Tests FFT, IFFT, and Hilbert implementations in `zdomain.hpp`.

Author

Samuel Andrew Wisner, awisner94@gmail.com

Definition in file `fft_test2.cpp`.

8.8.2 Function Documentation

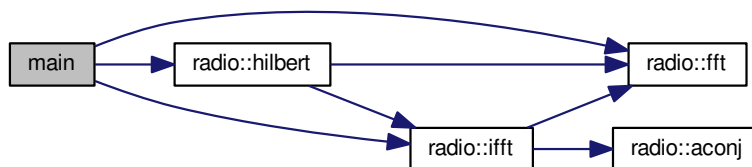
8.8.2.1 `int main ()`

This program tests the `fft()`, `ifft()`, and `hilbert()` functions in the `zdomain.hpp` file.

This code is based on code from http://rosettacode.org/wiki/Fast_Fourier_transform#C.2B.2B.

Definition at line 22 of file `fft_test2.cpp`.

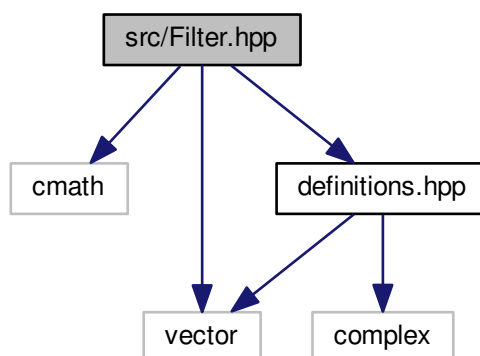
Here is the call graph for this function:



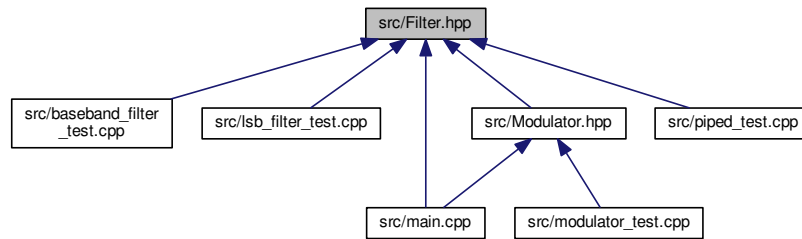
8.9 src/Filter.hpp File Reference

Defines the Filter class.

```
#include <cmath>
#include <vector>
#include "definitions.hpp"
Include dependency graph for Filter.hpp:
```



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



Classes

- class [radio::Filter](#)

Namespaces

- [radio](#)

contains helper-functions for [main\(\)](#)

8.9.1 Detailed Description

Defines the Filter class.

Author

Samuel Andrew Wisner, awisner94@gmail.com

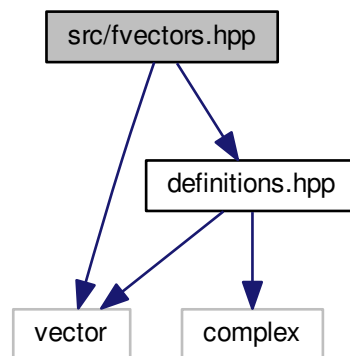
Definition in file [Filter.hpp](#).

8.10 src/fvectors.hpp File Reference

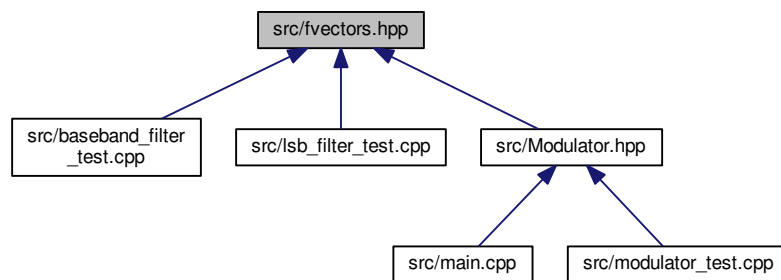
Defines the transfer function coefficients used in the instances of the Filter class in this program.

```
#include <vector>
#include "definitions.hpp"
```

Include dependency graph for fvectors.hpp:



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



Namespaces

- [radio](#)
contains helper-functions for [main\(\)](#)

Variables

- `fparams radio::F_BASEBAND`
- `fparams radio::F_LOWERSIDEBAND`
- `fparams radio::F_UPPERSIDEBAND`

8.10.1 Detailed Description

Defines the transfer function coefficients used in the instances of the Filter class in this program.

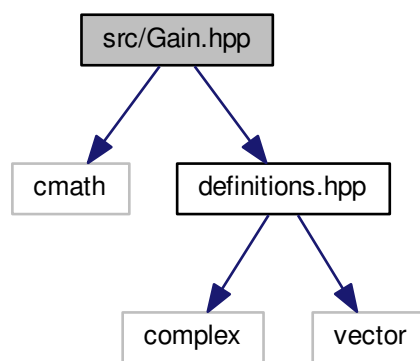
Author

Samuel Andrew Wisner, awisner94@gmail.com

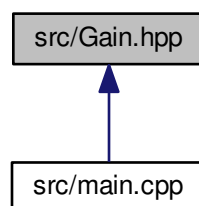
Definition in file [fvectors.hpp](#).

8.11 src/Gain.hpp File Reference

```
#include <cmath>
#include "definitions.hpp"
Include dependency graph for Gain.hpp:
```



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

**Classes**

- class [radio::Gain](#)

Namespaces

- [radio](#)

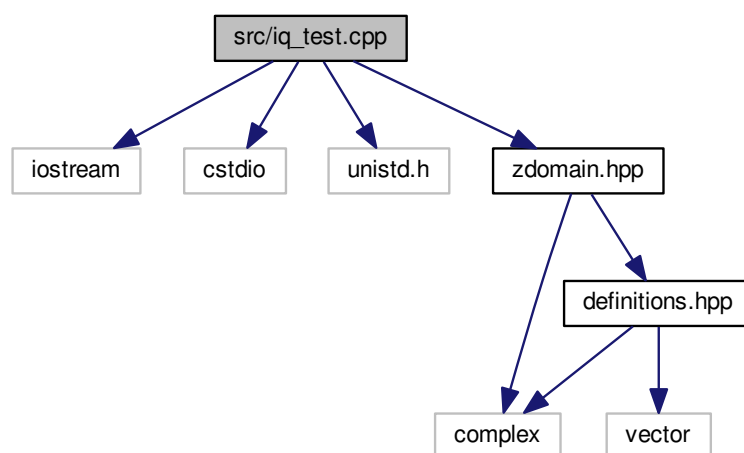
contains helper-functions for [main\(\)](#)

8.12 src/iq_test.cpp File Reference

Generates test IQ signal.

```
#include <iostream>
#include <cstdio>
#include <unistd.h>
#include "zdomain.hpp"
```

Include dependency graph for iq_test.cpp:



Functions

- int [main](#) ()

8.12.1 Detailed Description

Generates test IQ signal.

Author

Samuel Andrew Wisner, awisner94@gmail.com

Definition in file [iq_test.cpp](#).

8.12.2 Function Documentation

8.12.2.1 int main ()

This small program demonstrates the IQ generation abilities of the [makeIQ\(\)](#) function.

Definition at line 20 of file `iq_test.cpp`.

Here is the call graph for this function:



8.13 src/lsb_filter_test.cpp File Reference

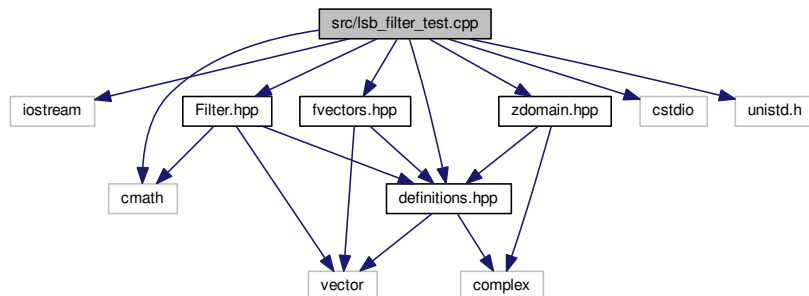
contains a program to test the LSB-via-filter implementation

```

#include <iostream>
#include <cmath>
#include <cstdio>
#include <unistd.h>
#include "definitions.hpp"
#include "Filter.hpp"
#include "fvectors.hpp"
#include "zdomain.hpp"

```

Include dependency graph for lsb_filter_test.cpp:



Functions

- int [main](#) ()

8.13.1 Detailed Description

contains a program to test the LSB-via-filter implementation

Author

Samuel Andrew Wisner, awisner94@gmail.com

Definition in file [lsb_filter_test.cpp](#).

8.13.2 Function Documentation

8.13.2.1 int main ()

Tests an implementation of LSB modulation through a filter.

Definition at line 23 of file lsb_filter_test.cpp.

Here is the call graph for this function:



8.14 src/main.cpp File Reference

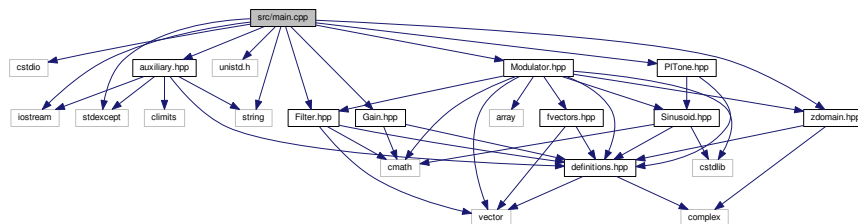
contains the "brains" of the entire project

```

#include <cstdio>
#include <iostream>
#include <stdexcept>
#include <string>
#include <unistd.h>
#include "auxiliary.hpp"
#include "Filter.hpp"
#include "Gain.hpp"
#include "Modulator.hpp"
#include "PlTone.hpp"
#include "zdomain.hpp"

```

Include dependency graph for main.cpp:



Functions

- int [main](#) (int argc, char *argv[])

8.14.1 Detailed Description

contains the "brains" of the entire project

Author

Samuel Andrew Wisner, awisner94@gmail.com

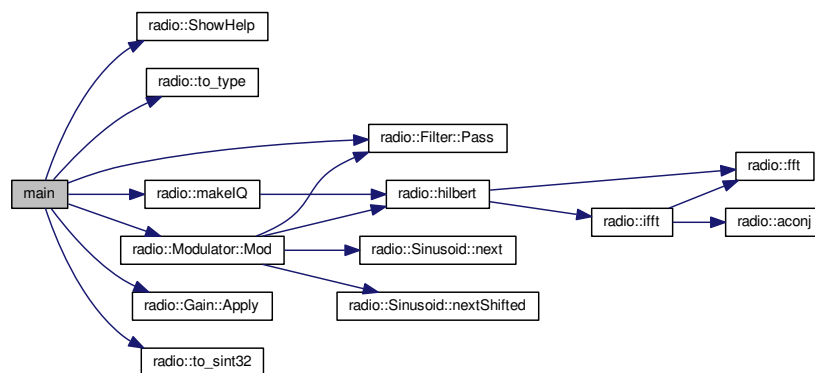
Definition in file [main.cpp](#).

8.14.2 Function Documentation**8.14.2.1 int main (int argc, char * argv[])**

Final result of the entire project. Completes all goals and more!

Definition at line 26 of file main.cpp.

Here is the call graph for this function:

**8.15 src/mic_test.cpp File Reference**

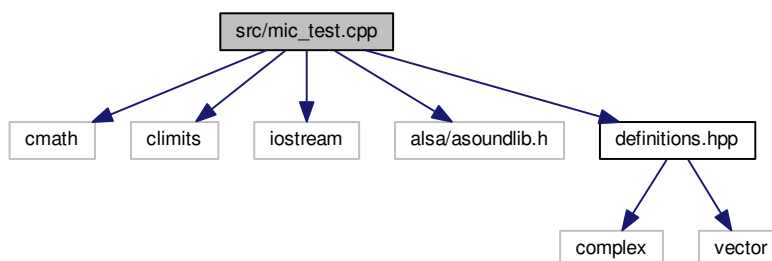
Tests getting mic input via ALSA May not even compile at the moment.

```

#include <cmath>
#include <climits>
#include <iostream>
#include <alsa/asoundlib.h>
#include "definitions.hpp"

```

Include dependency graph for mic_test.cpp:



Functions

- int [main](#) ()

8.15.1 Detailed Description

Tests getting mic input via ALSA May not even compile at the moment.

Author

Samuel Andrew Wisner, awisner94@gmail.com

Definition in file [mic_test.cpp](#).

8.15.2 Function Documentation

8.15.2.1 int main ()

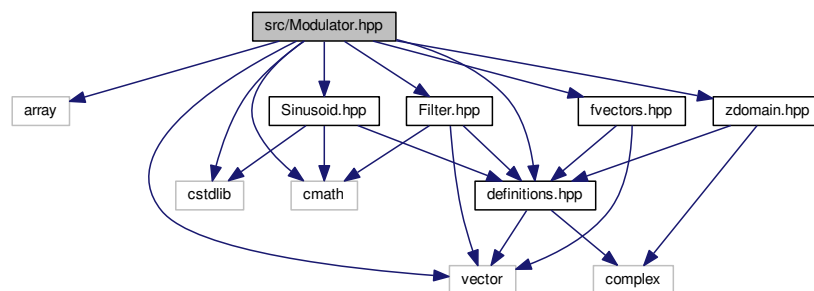
This program tests taking information from the microphone via the ALSA API. Not sure if it works.

Definition at line 21 of file mic_test.cpp.

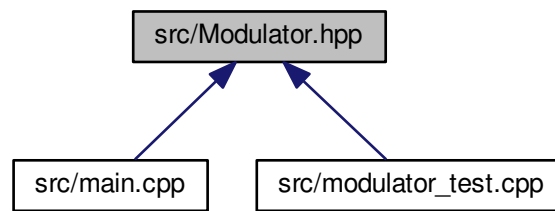
8.16 src/Modulator.hpp File Reference

```
#include <array>
#include <cmath>
#include <cstdlib>
#include <vector>
#include "definitions.hpp"
#include "Filter.hpp"
#include "fvectors.hpp"
#include "Sinusoid.hpp"
#include "zdomain.hpp"
```

Include dependency graph for Modulator.hpp:



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



Classes

- class [radio::Modulator](#)

Namespaces

- [radio](#)
contains helper-functions for *main()*

Variables

- const [uint32](#) [radio::FREQ_INTERMEDIATE](#) = 20000
- const [uint32](#) [radio::SAMPLING_RATE](#) = 48000

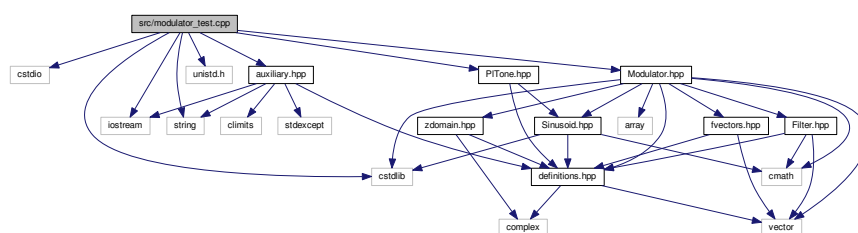
8.17 src/modulator_test.cpp File Reference

```

#include <cstdio>
#include <cstdlib>
#include <iostream>
#include <string>
#include <unistd.h>
#include "auxiliary.hpp"
#include "Modulator.hpp"
#include "PlTone.hpp"

```

Include dependency graph for modulator_test.cpp:



Functions

- int [main](#) (int argc, char *argv[])

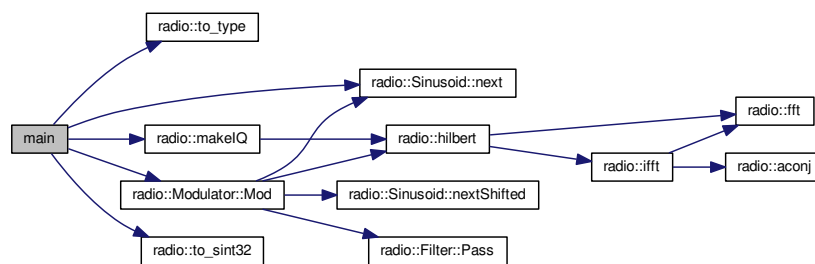
8.17.1 Function Documentation

8.17.1.1 int main (int argc, char * argv[])

Program to test the Modulator class with a self-generated sinusoidal input.

Definition at line 22 of file modulator_test.cpp.

Here is the call graph for this function:



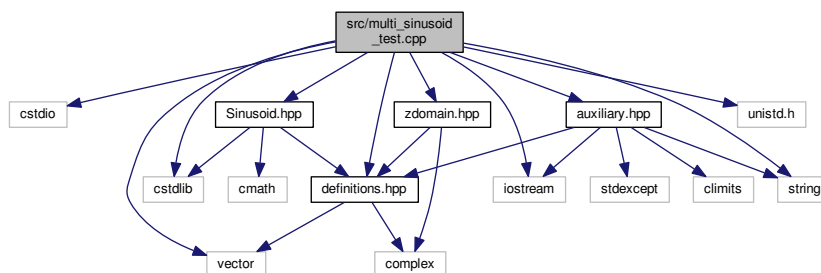
8.18 src/multi_sinusoid_test.cpp File Reference

```

#include <cstdio>
#include <cstdlib>
#include <iostream>
#include <string>
#include <unistd.h>
#include <vector>
#include "auxiliary.hpp"
#include "definitions.hpp"
#include "Sinusoid.hpp"
#include "zdomain.hpp"

```

Include dependency graph for multi_sinusoid_test.cpp:



Functions

- int [main](#) (int argc, char *argv[])

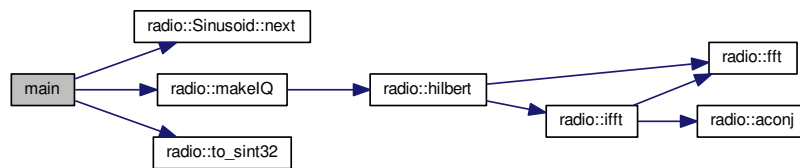
8.18.1 Function Documentation

8.18.1.1 int main (int argc, char * argv[])

Program to test the Sinusoid class and demonstrate the frequency range of the sound card.

Definition at line 26 of file multi_sinusoid_test.cpp.

Here is the call graph for this function:



8.19 src/piped_test.cpp File Reference

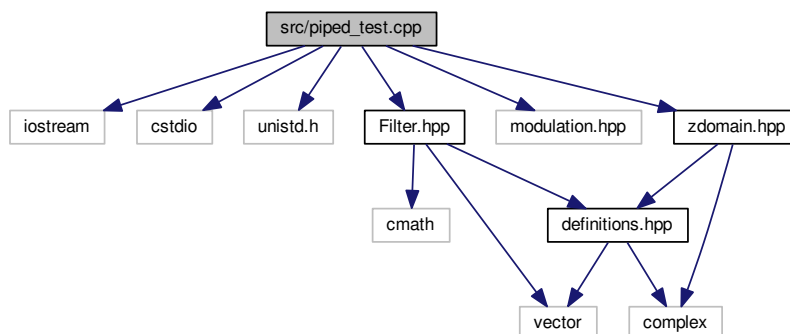
contains the original program used to test the piping-in idea

```

#include <iostream>
#include <cstdio>
#include <unistd.h>
#include "Filter.hpp"
#include "modulation.hpp"
#include "zdomain.hpp"

```

Include dependency graph for `piped_test.cpp`:



Functions

- int [main](#) ()

8.19.1 Detailed Description

contains the original program used to test the piping-in idea

Author

Samuel Andrew Wisner, awisner94@gmail.com

Definition in file [piped_test.cpp](#).

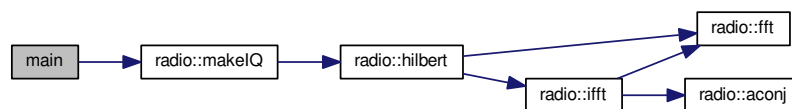
8.19.2 Function Documentation

8.19.2.1 int main ()

Program originally used to test whether baseband audio could be piped into the program in real time.

Definition at line 22 of file [piped_test.cpp](#).

Here is the call graph for this function:



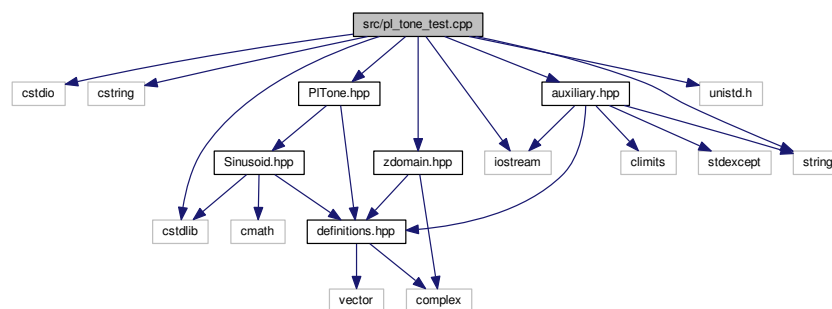
8.20 src/pl_tone_test.cpp File Reference

```

#include <cstdio>
#include <cstring>
#include <cstdlib>
#include <iostream>
#include <string>
#include <unistd.h>
#include "auxiliary.hpp"
#include "PlTone.hpp"
#include "zdomain.hpp"

```

Include dependency graph for `pl_tone_test.cpp`:



Functions

- int [main](#) (int argc, char *argv[])

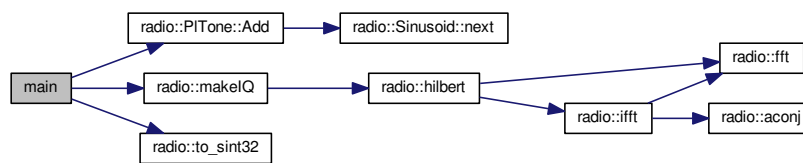
8.20.1 Function Documentation

8.20.1.1 int main (int argc, char * argv[])

Program to test the PITone class.

Definition at line 23 of file pl_tone_test.cpp.

Here is the call graph for this function:



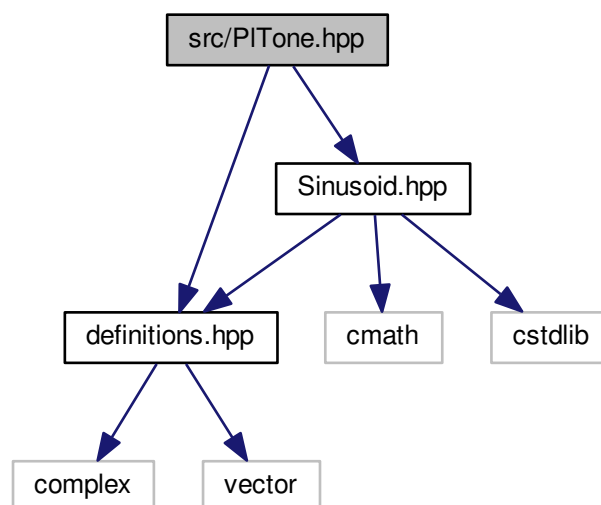
8.21 src/PITone.hpp File Reference

contains the PITone class

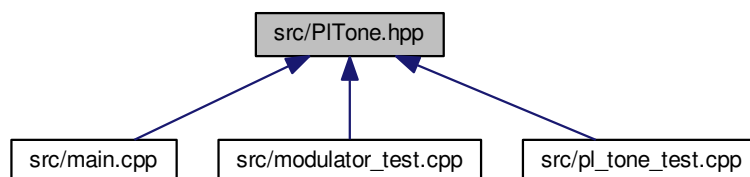
```
#include "definitions.hpp"
```

```
#include "Sinusoid.hpp"
```

Include dependency graph for PITone.hpp:



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



Classes

- class [radio::PITone](#)

Namespaces

- [radio](#)

contains helper-functions for [main\(\)](#)

8.21.1 Detailed Description

contains the PITone class

Author

Samuel Andrew Wisner, awisner94@gmail.com

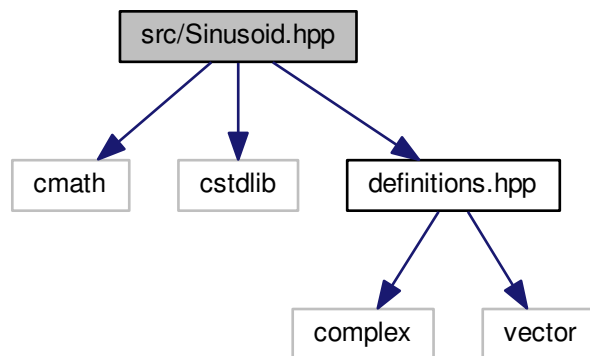
Definition in file [PITone.hpp](#).

8.22 src/Sinusoid.hpp File Reference

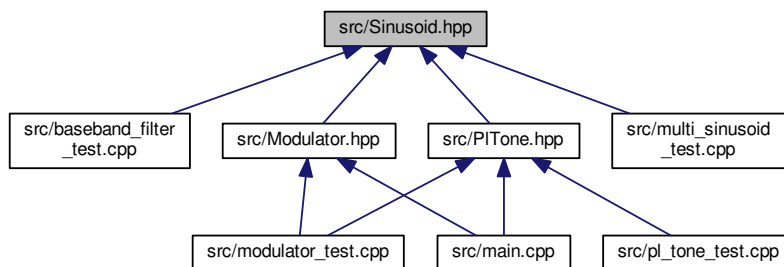
contains the Sinusoid class

```
#include <cmath>
#include <cstdlib>
#include "definitions.hpp"
```

Include dependency graph for Sinusoid.hpp:



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



Classes

- class [radio::Sinusoid](#)

Namespaces

- [radio](#)
contains helper-functions for [main\(\)](#)

8.22.1 Detailed Description

contains the Sinusoid class

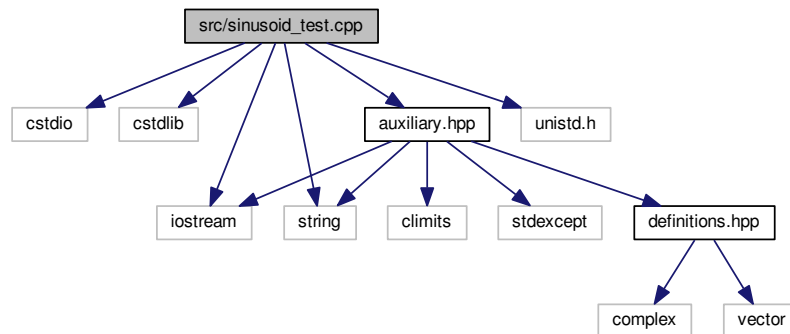
Author

Samuel Andrew Wisner, awisner94@gmail.com

Definition in file [Sinusoid.hpp](#).

8.23 src/sinusoid_test.cpp File Reference

```
#include <cstdio>
#include <cstdlib>
#include <iostream>
#include <string>
#include <unistd.h>
#include "auxiliary.hpp"
Include dependency graph for sinusoid_test.cpp:
```



Functions

- int [main](#) (int argc, char *argv[])

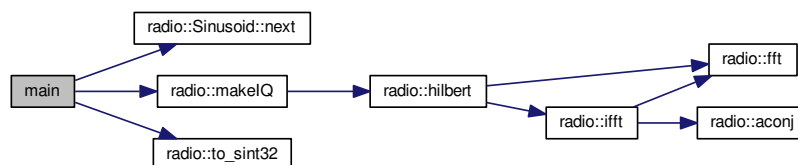
8.23.1 Function Documentation

8.23.1.1 int main (int argc, char * argv[])

Program to test the Sinusoid class.

Definition at line 20 of file sinusoid_test.cpp.

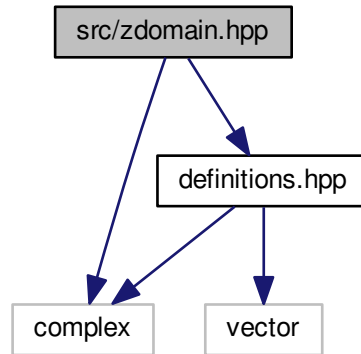
Here is the call graph for this function:



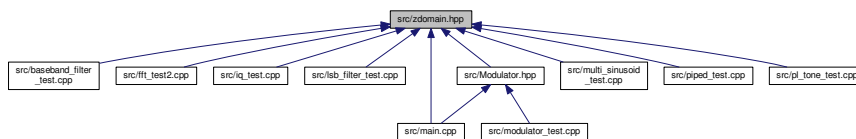
8.24 src/zdomain.hpp File Reference

Contains the functions to manipulate sequential data in the frequency (z) domain.

```
#include <complex>
#include "definitions.hpp"
Include dependency graph for zdomain.hpp:
```



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



Namespaces

- [radio](#)
contains helper-functions for [main\(\)](#)

Functions

- void [radio::aconj](#) ([cfloat32](#) *data, [uint32](#) size)
- void [radio::fft](#) ([cfloat32](#) *data, [uint32](#) size)
- void [radio::hilbert](#) ([float32](#) *data, [float32](#) *dest, [uint32](#) size)
- void [radio::ifft](#) ([cfloat32](#) *data, [uint32](#) size)
- void [radio::makeIQ](#) ([float32](#) *data, [float32](#) *dest, [uint32](#) size)

8.24.1 Detailed Description

Contains the functions to manipulate sequential data in the frequency (z) domain.

Author

Samuel Andrew Wisner, awisner94@gmail.com

Definition in file [zdomain.hpp](#).

Index

aconj
radio, [13](#)

Age
radio, [12](#)

Argument
radio, [12](#)

DEN
radio, [12](#)

DSB_LC
radio, [13](#)

DSB_SC
radio, [13](#)

FM_NARROW
radio, [13](#)

FM_WIDE
radio, [13](#)

FREQ
radio, [12](#)

fft
radio, [13](#)

Fractional
radio, [12](#)

hilbert
radio, [14](#)

ifft
radio, [15](#)

LSB_FILTERED
radio, [13](#)

LSB_HILBERT
radio, [13](#)

MODE
radio, [12](#)

makefile, [31](#)

NEW
radio, [12](#)

NUM
radio, [12](#)

OLD
radio, [12](#)

PL_TONE
radio, [12](#)

radio, [11](#)

aconj, [13](#)

Age, [12](#)

Argument, [12](#)

DEN, [12](#)

DSB_LC, [13](#)

DSB_SC, [13](#)

FM_NARROW, [13](#)

FM_WIDE, [13](#)

FREQ, [12](#)

fft, [13](#)

Fractional, [12](#)

hilbert, [14](#)

ifft, [15](#)

LSB_FILTERED, [13](#)

LSB_HILBERT, [13](#)

MODE, [12](#)

NEW, [12](#)

NUM, [12](#)

OLD, [12](#)

PL_TONE, [12](#)

USB_FILTERED, [13](#)

USB_HILBERT, [13](#)

USB_FILTERED
radio, [13](#)

USB_HILBERT
radio, [13](#)