

iir1

Generated by Doxygen 1.8.13

Contents

1	IIR1 -- Realtime C++ filter library	2
2	Namespace Index	4
2.1	Namespace List	4
3	Hierarchical Index	5
3.1	Class Hierarchy	5
4	Class Index	9
4.1	Class List	9
5	Namespace Documentation	12
5.1	lir Namespace Reference	12
5.1.1	Detailed Description	13
5.1.2	Enumeration Type Documentation	14
5.1.3	Function Documentation	14
5.2	lir::Bessel Namespace Reference	15
5.2.1	Detailed Description	16
5.3	lir::Butterworth Namespace Reference	16
5.3.1	Detailed Description	16
5.4	lir::ChebyshevI Namespace Reference	16
5.4.1	Detailed Description	17
5.5	lir::ChebyshevII Namespace Reference	17
5.5.1	Detailed Description	17
5.6	lir::RBJ Namespace Reference	17
5.6.1	Detailed Description	18

6	Class Documentation	18
6.1	lir::RBJ::AllPass Struct Reference	18
6.2	lir::Bessel::AnalogLowPass Class Reference	18
6.3	lir::ChebyshevI::AnalogLowPass Class Reference	19
6.4	lir::Butterworth::AnalogLowPass Class Reference	19
6.4.1	Detailed Description	19
6.5	lir::ChebyshevII::AnalogLowPass Class Reference	19
6.6	lir::Bessel::AnalogLowShelf Class Reference	20
6.7	lir::Butterworth::AnalogLowShelf Class Reference	20
6.8	lir::ChebyshevI::AnalogLowShelf Class Reference	20
6.9	lir::ChebyshevII::AnalogLowShelf Class Reference	21
6.10	lir::RootFinderBase::Array Struct Reference	21
6.11	lir::ChebyshevII::BandPass< MaxOrder, StateType > Struct Template Reference	21
6.11.1	Detailed Description	21
6.11.2	Member Function Documentation	22
6.12	lir::Bessel::BandPass< MaxOrder, StateType > Struct Template Reference	22
6.12.1	Detailed Description	22
6.12.2	Member Function Documentation	23
6.13	lir::ChebyshevI::BandPass< MaxOrder, StateType > Struct Template Reference	23
6.13.1	Detailed Description	23
6.13.2	Member Function Documentation	24
6.14	lir::Butterworth::BandPass< MaxOrder, StateType > Struct Template Reference	24
6.14.1	Detailed Description	24
6.14.2	Member Function Documentation	25
6.15	lir::RBJ::BandPass1 Struct Reference	25
6.15.1	Detailed Description	25
6.15.2	Member Function Documentation	26
6.16	lir::RBJ::BandPass2 Struct Reference	26
6.16.1	Detailed Description	26
6.16.2	Member Function Documentation	27

6.17	lir::Butterworth::BandPassBase Struct Reference	27
6.18	lir::ChebyshevI::BandPassBase Struct Reference	28
6.19	lir::Bessel::BandPassBase Struct Reference	28
6.20	lir::ChebyshevII::BandPassBase Struct Reference	28
6.21	lir::BandPassTransform Class Reference	29
6.21.1	Detailed Description	29
6.22	lir::Butterworth::BandShelf< MaxOrder, StateType > Struct Template Reference	29
6.22.1	Detailed Description	30
6.22.2	Member Function Documentation	30
6.23	lir::ChebyshevII::BandShelf< MaxOrder, StateType > Struct Template Reference	30
6.23.1	Detailed Description	31
6.23.2	Member Function Documentation	31
6.24	lir::RBJ::BandShelf Struct Reference	31
6.25	lir::ChebyshevI::BandShelf< MaxOrder, StateType > Struct Template Reference	32
6.25.1	Detailed Description	32
6.25.2	Member Function Documentation	32
6.26	lir::ChebyshevI::BandShelfBase Struct Reference	33
6.27	lir::Butterworth::BandShelfBase Struct Reference	33
6.28	lir::ChebyshevII::BandShelfBase Struct Reference	34
6.29	lir::ChebyshevII::BandStop< MaxOrder, StateType > Struct Template Reference	34
6.29.1	Detailed Description	34
6.29.2	Member Function Documentation	34
6.30	lir::RBJ::BandStop Struct Reference	35
6.30.1	Detailed Description	35
6.30.2	Member Function Documentation	35
6.31	lir::ChebyshevI::BandStop< MaxOrder, StateType > Struct Template Reference	36
6.31.1	Detailed Description	36
6.31.2	Member Function Documentation	36
6.32	lir::Bessel::BandStop< MaxOrder, StateType > Struct Template Reference	37
6.32.1	Detailed Description	37

6.32.2	Member Function Documentation	37
6.33	<code>lir::Butterworth::BandStop< MaxOrder, StateType ></code> Struct Template Reference	38
6.33.1	Detailed Description	38
6.33.2	Member Function Documentation	38
6.34	<code>lir::Bessel::BandStopBase</code> Struct Reference	39
6.35	<code>lir::ChebyshevI::BandStopBase</code> Struct Reference	39
6.36	<code>lir::Butterworth::BandStopBase</code> Struct Reference	40
6.37	<code>lir::ChebyshevII::BandStopBase</code> Struct Reference	40
6.38	<code>lir::BandStopTransform</code> Class Reference	40
6.38.1	Detailed Description	40
6.39	<code>lir::Biquad</code> Class Reference	41
6.40	<code>lir::BiquadBase</code> Class Reference	41
6.41	<code>lir::BiquadPoleState</code> Struct Reference	42
6.42	<code>lir::Cascade</code> Class Reference	42
6.43	<code>lir::CascadeStages< MaxStages, StateType ></code> Class Template Reference	43
6.44	<code>lir::ComplexPair</code> Struct Reference	43
6.44.1	Detailed Description	43
6.45	<code>lir::DirectFormI</code> Class Reference	43
6.45.1	Detailed Description	44
6.46	<code>lir::DirectFormII</code> Class Reference	44
6.46.1	Detailed Description	44
6.47	<code>lir::EnvelopeFollower< Channels, Value ></code> Class Template Reference	44
6.48	<code>lir::ChebyshevII::HighPass< MaxOrder, StateType ></code> Struct Template Reference	44
6.48.1	Detailed Description	45
6.48.2	Member Function Documentation	45
6.49	<code>lir::Butterworth::HighPass< MaxOrder, StateType ></code> Struct Template Reference	45
6.49.1	Detailed Description	46
6.49.2	Member Function Documentation	46
6.50	<code>lir::RBJ::HighPass</code> Struct Reference	46
6.50.1	Detailed Description	47

6.50.2	Member Function Documentation	47
6.51	lir::Bessel::HighPass< MaxOrder, StateType > Struct Template Reference	47
6.51.1	Detailed Description	48
6.51.2	Member Function Documentation	48
6.52	lir::ChebyshevI::HighPass< MaxOrder, StateType > Struct Template Reference	48
6.52.1	Detailed Description	49
6.52.2	Member Function Documentation	49
6.53	lir::ChebyshevI::HighPassBase Struct Reference	49
6.54	lir::Butterworth::HighPassBase Struct Reference	50
6.55	lir::Bessel::HighPassBase Struct Reference	50
6.56	lir::ChebyshevII::HighPassBase Struct Reference	50
6.57	lir::HighPassTransform Class Reference	51
6.57.1	Detailed Description	51
6.58	lir::Butterworth::HighShelf< MaxOrder, StateType > Struct Template Reference	51
6.58.1	Detailed Description	52
6.58.2	Member Function Documentation	52
6.59	lir::ChebyshevII::HighShelf< MaxOrder, StateType > Struct Template Reference	52
6.59.1	Detailed Description	53
6.59.2	Member Function Documentation	53
6.60	lir::RBJ::HighShelf Struct Reference	53
6.61	lir::ChebyshevI::HighShelf< MaxOrder, StateType > Struct Template Reference	54
6.61.1	Detailed Description	54
6.61.2	Member Function Documentation	54
6.62	lir::ChebyshevI::HighShelfBase Struct Reference	55
6.63	lir::Butterworth::HighShelfBase Struct Reference	55
6.64	lir::ChebyshevII::HighShelfBase Struct Reference	55
6.65	lir::RBJ::IIRNotch Struct Reference	56
6.65.1	Detailed Description	56
6.65.2	Member Function Documentation	56
6.66	lir::Layout< MaxPoles > Class Template Reference	57

6.66.1 Detailed Description	57
6.67 lir::LayoutBase Class Reference	57
6.67.1 Detailed Description	58
6.68 lir::ChebyshevI::LowPass< MaxOrder, StateType > Struct Template Reference	58
6.68.1 Detailed Description	58
6.68.2 Member Function Documentation	58
6.69 lir::Bessel::LowPass< MaxOrder, StateType > Struct Template Reference	59
6.69.1 Detailed Description	59
6.69.2 Member Function Documentation	59
6.70 lir::RBJ::LowPass Struct Reference	60
6.70.1 Detailed Description	60
6.70.2 Member Function Documentation	60
6.71 lir::Butterworth::LowPass< MaxOrder, StateType > Struct Template Reference	61
6.71.1 Detailed Description	61
6.71.2 Member Function Documentation	61
6.72 lir::ChebyshevII::LowPass< MaxOrder, StateType > Struct Template Reference	62
6.72.1 Detailed Description	62
6.72.2 Member Function Documentation	62
6.73 lir::Bessel::LowPassBase Struct Reference	63
6.74 lir::Butterworth::LowPassBase Struct Reference	63
6.75 lir::ChebyshevI::LowPassBase Struct Reference	64
6.76 lir::ChebyshevII::LowPassBase Struct Reference	64
6.77 lir::LowPassTransform Class Reference	64
6.77.1 Detailed Description	65
6.78 lir::Butterworth::LowShelf< MaxOrder, StateType > Struct Template Reference	65
6.78.1 Detailed Description	65
6.78.2 Member Function Documentation	65
6.79 lir::ChebyshevII::LowShelf< MaxOrder, StateType > Struct Template Reference	66
6.79.1 Detailed Description	66
6.79.2 Member Function Documentation	66

6.80	lir::RBJ::LowShelf Struct Reference	67
6.81	lir::ChebyshevI::LowShelf< MaxOrder, StateType > Struct Template Reference	67
6.81.1	Detailed Description	68
6.81.2	Member Function Documentation	68
6.82	lir::Bessel::LowShelfBase Struct Reference	68
6.83	lir::ChebyshevI::LowShelfBase Struct Reference	69
6.84	lir::Butterworth::LowShelfBase Struct Reference	69
6.85	lir::ChebyshevII::LowShelfBase Struct Reference	69
6.86	lir::Custom::OnePole Struct Reference	70
6.87	lir::PoleFilter< BaseClass, StateType, MaxAnalogPoles, MaxDigitalPoles > Struct Template Reference	70
6.87.1	Detailed Description	71
6.88	lir::PoleFilterBase< AnalogPrototype > Class Template Reference	71
6.88.1	Detailed Description	71
6.89	lir::PoleFilterBase2 Class Reference	72
6.89.1	Detailed Description	72
6.90	lir::PoleZeroPair Struct Reference	72
6.90.1	Detailed Description	73
6.91	lir::RBJ::RBJbase Struct Reference	73
6.91.1	Detailed Description	74
6.92	lir::RootFinder< maxdegree > Struct Template Reference	74
6.93	lir::RootFinderBase Class Reference	74
6.93.1	Detailed Description	74
6.94	lir::SlopeDetector< Channels, Value > Class Template Reference	75
6.95	lir::Cascade::Stage Struct Reference	75
6.96	lir::BiquadBase::State< StateType > Struct Template Reference	75
6.97	lir::Cascade::Storage Struct Reference	75
6.98	lir::TransposedDirectFormII Class Reference	76
6.99	lir::Custom::TwoPole Struct Reference	76
6.100	lir::Bessel::Workspace< MaxOrder > Struct Template Reference	76
6.101	lir::Bessel::WorkspaceBase Struct Reference	76

1 IIR1 -- Realtime C++ filter library

An infinite impulse response (IIR) filter library for Linux, Mac OSX and Windows which implements Bessel, Butterworth, RBJ and Chebychev filters.

The filter processes the data sample by sample for realtime processing.

It uses templates to allocate the required memory so that it can run without any malloc / new commands for example on embedded systems.

How to use the filter

First the filter is instantiated, then the parameters are set with the function `setup` and then it's ready to be used for sample by sample realtime filtering.

Setting the filter parameters

All filters are available as lowpass, highpass, bandpass and bandstop/notch filters. Butterworth / Chebyshev offer also low/high/band-shelves with specified passband gain and 0dB gain in the stopband.

See the header files in `\iir` or the documentation for the arguments of the `setup` commands.

The examples below are for lowpass filters:

1. Butterworth – [Butterworth.h](#) Standard filter suitable for most applications. Monotonic response.

```
const int order = 4; // 4th order (=2 biquads)
Iir::Butterworth::LowPass<order> f;
const float samplingrate = 1000; // Hz
const float cutoff_frequency = 5; // Hz
f.setup (samplingrate, cutoff_frequency);
```

2. Chebyshev Type I – [ChebyshevI.h](#) With permissible passband ripple in dB.

```
Iir::ChebyshevI::LowPass<order> f;
const float passband_ripple_in_db = 5;
f.setup (samplingrate,
        cutoff_frequency,
        passband_ripple_in_db);
```

3. Chebyshev Type II – [ChebyshevII.h](#) With worst permissible stopband rejection in dB.

```
Iir::ChebyshevII::LowPass<order> f;
double stopband_ripple_in_db = 20;
f.setup (samplingrate,
        cutoff_frequency,
        stopband_ripple_in_db);
```

4. RBJ – [RBJ.h](#) 2nd order filters with cutoff and Q factor.

```
Iir::RBJ::LowPass f;
const float cutoff_frequency = 100;
const float Q_factor = 5;
f.setup (samplingrate, cutoff_frequency, Q_factor);
```

Realtime filtering sample by sample

Samples are processed one by one. In the example below a sample `x` is processed with the `filter` command and then saved in `y`. The type of `x` can either be float or double:

```
float y = f.filter(x);
```

This is then repeated for every incoming sample in a loop or event handler.

Packages for Ubuntu (xenial / bionic):

If you have Ubuntu xenial or bionic then install it as a pre-compiled package:

```
sudo add-apt-repository ppa:berndporr/usbdx
```

It's available for 32,64 bit PC and 32,64 bit ARM (Raspberry PI etc)

Compilation from source

The build tool is `cmake` which generates the make- or project files for the different platforms. `cmake` is available for Linux, Windows and Mac. It also compiles directly on a Raspberry PI.

Linux / Mac

Run

```
cmake .
```

which generates the Makefile. Then run:

```
make  
sudo make install
```

which installs it under `/usr/local/lib` and `/usr/local/include`.

Both gcc and clang have been tested.

Windows

```
cmake -G "Visual Studio 15 2017 Win64" .
```

See `cmake` for the different build-options. Above is for a 64 bit build. Then start Visual C++ and open the solution. This will create the DLL and the LIB files. Under Windows it's highly recommended to use the static library and link it into the application program.

Unit tests

Run unit tests by typing `make test` or just `ctest`. These test if after a delta pulse all filters relax to zero and that their outputs never become NaN.

Documentation

Overview

For an overview of the class structure and general concepts have a look at `Documentation.txt`.

Learn from the demos

The easiest way to learn is from the examples which are in the `demo` directory. A delta pulse as a test signal is sent into the different filters and saved in a file. With the Python script `plot_impulse_fresponse.py` you can then plot the frequency responses.

Also the directory containing the unit tests provides examples for every filter type.

Detailed documentation

A PDF of all classes, methods and in particular `setup` functions is in the `doc/pdf` directory.

Run `doxygen` to generate the HTML documentation.

Example filter responses

These are the filter responses generated by `iirdemo.cpp` in the `/demo/` directory and then plotted with `plot_impulse_fresponse.py`.

Credits

This library has been adapted from Vinnie Falco's original work which can be found here:

<https://github.com/vinniefalco/DSPFilters>

While his original library processes audio arrays this library has been adapted to do realtime processing sample by sample. Also, in contrast to the original library the `setup` command won't require the filter order and instead remembers it from the template argument.

Enjoy!

Bernd Porr – <http://www.berndporr.me.uk>

2 Namespace Index

2.1 Namespace List

Here is a list of all documented namespaces with brief descriptions:

lir	12
lir::Bessel	15
lir::Butterworth	16
lir::ChebyshevI	16
lir::ChebyshevII	17
lir::RBJ	17

3 Hierarchical Index

3.1 Class Hierarchy

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

lir::RootFinderBase::Array	21
BandPassBase	
lir::PoleFilter< BandPassBase, StateType, MaxOrder, MaxOrder *2 >	70
lir::Bessel::BandPass< MaxOrder, StateType >	22
lir::Butterworth::BandPass< MaxOrder, StateType >	24
lir::ChebyshevI::BandPass< MaxOrder, StateType >	23
lir::ChebyshevII::BandPass< MaxOrder, StateType >	21
lir::BandPassTransform	29
BandShelfBase	
lir::PoleFilter< BandShelfBase, StateType, MaxOrder, MaxOrder *2 >	70
lir::Butterworth::BandShelf< MaxOrder, StateType >	29
lir::ChebyshevI::BandShelf< MaxOrder, StateType >	32
lir::ChebyshevII::BandShelf< MaxOrder, StateType >	30
BandStopBase	
lir::PoleFilter< BandStopBase, StateType, MaxOrder, MaxOrder *2 >	70
lir::Bessel::BandStop< MaxOrder, StateType >	37
lir::Butterworth::BandStop< MaxOrder, StateType >	38
lir::ChebyshevI::BandStop< MaxOrder, StateType >	36
lir::ChebyshevII::BandStop< MaxOrder, StateType >	34
lir::BandStopTransform	40
lir::BiquadBase	41
lir::Biquad	41

lir::Cascade::Stage	75
lir::Custom::OnePole	70
lir::Custom::TwoPole	76
lir::RBJ::RBJbase	73
lir::RBJ::AllPass	18
lir::RBJ::BandPass1	25
lir::RBJ::BandPass2	26
lir::RBJ::BandShelf	31
lir::RBJ::BandStop	35
lir::RBJ::HighPass	46
lir::RBJ::HighShelf	53
lir::RBJ::IIRNotch	56
lir::RBJ::LowPass	60
lir::RBJ::LowShelf	67
lir::Cascade	42
lir::PoleFilterBase2	72
lir::PoleFilterBase< AnalogPrototype >	71
lir::PoleFilterBase< AnalogLowPass >	71
lir::Bessel::BandPassBase	28
lir::Bessel::BandStopBase	39
lir::Bessel::HighPassBase	50
lir::Bessel::LowPassBase	63
lir::Butterworth::BandPassBase	27
lir::Butterworth::BandStopBase	40
lir::Butterworth::HighPassBase	50
lir::Butterworth::LowPassBase	63
lir::ChebyshevI::BandPassBase	28
lir::ChebyshevI::BandStopBase	39
lir::ChebyshevI::HighPassBase	49
lir::ChebyshevI::LowPassBase	64
lir::ChebyshevII::BandPassBase	28
lir::ChebyshevII::BandStopBase	40

lir::ChebyshevII::HighPassBase	50
lir::ChebyshevII::LowPassBase	64
lir::PoleFilterBase< AnalogLowShelf >	71
lir::Bessel::LowShelfBase	68
lir::Butterworth::BandShelfBase	33
lir::Butterworth::HighShelfBase	55
lir::Butterworth::LowShelfBase	69
lir::ChebyshevI::BandShelfBase	33
lir::ChebyshevI::HighShelfBase	55
lir::ChebyshevI::LowShelfBase	69
lir::ChebyshevII::BandShelfBase	34
lir::ChebyshevII::HighShelfBase	55
lir::ChebyshevII::LowShelfBase	69
lir::CascadeStages< MaxStages, StateType >	43
lir::CascadeStages<(MaxAnalogPoles+1)/2, StateType >	43
lir::PoleFilter< HighPassBase, StateType, MaxOrder >	70
lir::Bessel::HighPass< MaxOrder, StateType >	47
lir::Butterworth::HighPass< MaxOrder, StateType >	45
lir::ChebyshevI::HighPass< MaxOrder, StateType >	48
lir::ChebyshevII::HighPass< MaxOrder, StateType >	44
lir::PoleFilter< HighShelfBase, StateType, MaxOrder >	70
lir::Butterworth::HighShelf< MaxOrder, StateType >	51
lir::ChebyshevI::HighShelf< MaxOrder, StateType >	54
lir::ChebyshevII::HighShelf< MaxOrder, StateType >	52
lir::PoleFilter< LowPassBase, StateType, MaxOrder >	70
lir::Bessel::LowPass< MaxOrder, StateType >	59
lir::Butterworth::LowPass< MaxOrder, StateType >	61
lir::ChebyshevI::LowPass< MaxOrder, StateType >	58
lir::ChebyshevII::LowPass< MaxOrder, StateType >	62
lir::PoleFilter< LowShelfBase, StateType, MaxOrder >	70
lir::Butterworth::LowShelf< MaxOrder, StateType >	65
lir::ChebyshevI::LowShelf< MaxOrder, StateType >	67

lir::ChebyshevII::LowShelf < MaxOrder , StateType >	66
lir::CascadeStages <(MaxDigitalPoles+1)/2, StateType >	43
lir::PoleFilter < BaseClass , StateType , MaxAnalogPoles , MaxDigitalPoles >	70
lir::PoleFilter < BandPassBase , StateType , MaxOrder , MaxOrder *2 >	70
lir::PoleFilter < BandShelfBase , StateType , MaxOrder , MaxOrder *2 >	70
lir::PoleFilter < BandStopBase , StateType , MaxOrder , MaxOrder *2 >	70
complex_pair_t	
lir::ComplexPair	43
lir::DirectFormI	43
lir::DirectFormII	44
lir::EnvelopeFollower < Channels , Value >	44
HighPassBase	
lir::PoleFilter < HighPassBase , StateType , MaxOrder >	70
lir::HighPassTransform	51
HighShelfBase	
lir::PoleFilter < HighShelfBase , StateType , MaxOrder >	70
lir::Layout < MaxPoles >	57
lir::Layout < MaxAnalogPoles >	57
lir::Layout < MaxDigitalPoles >	57
lir::LayoutBase	57
lir::Bessel::AnalogLowPass	18
lir::Bessel::AnalogLowShelf	20
lir::Butterworth::AnalogLowPass	19
lir::Butterworth::AnalogLowShelf	20
lir::ChebyshevI::AnalogLowPass	19
lir::ChebyshevI::AnalogLowShelf	20
lir::ChebyshevII::AnalogLowPass	19
lir::ChebyshevII::AnalogLowShelf	21
LowPassBase	
lir::PoleFilter < LowPassBase , StateType , MaxOrder >	70
lir::LowPassTransform	64
LowShelfBase	
lir::PoleFilter < LowShelfBase , StateType , MaxOrder >	70
lir::PoleZeroPair	72

lir::BiquadPoleState	42
lir::RootFinderBase	74
lir::RootFinder< maxdegree >	74
lir::RootFinder< MaxOrder >	74
lir::SlopeDetector< Channels, Value > StateType	75
lir::BiquadBase::State< StateType >	75
lir::Cascade::Storage	75
lir::TransposedDirectFormII	76
lir::Bessel::WorkspaceBase	76
lir::Bessel::Workspace< MaxOrder > BaseClass	76
lir::PoleFilter< BaseClass, StateType, MaxAnalogPoles, MaxDigitalPoles >	70

4 Class Index

4.1 Class List

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

lir::RBJ::AllPass	18
lir::Bessel::AnalogLowPass	18
lir::ChebyshevI::AnalogLowPass	19
lir::Butterworth::AnalogLowPass	19
lir::ChebyshevII::AnalogLowPass	19
lir::Bessel::AnalogLowShelf	20
lir::Butterworth::AnalogLowShelf	20
lir::ChebyshevI::AnalogLowShelf	20
lir::ChebyshevII::AnalogLowShelf	21
lir::RootFinderBase::Array	21
lir::ChebyshevII::BandPass< MaxOrder, StateType >	21
lir::Bessel::BandPass< MaxOrder, StateType >	22
lir::ChebyshevI::BandPass< MaxOrder, StateType >	23
lir::Butterworth::BandPass< MaxOrder, StateType >	24
lir::RBJ::BandPass1	25

lir::RBJ::BandPass2	26
lir::Butterworth::BandPassBase	27
lir::ChebyshevI::BandPassBase	28
lir::Bessel::BandPassBase	28
lir::ChebyshevII::BandPassBase	28
lir::BandPassTransform	29
lir::Butterworth::BandShelf< MaxOrder, StateType >	29
lir::ChebyshevII::BandShelf< MaxOrder, StateType >	30
lir::RBJ::BandShelf	31
lir::ChebyshevI::BandShelf< MaxOrder, StateType >	32
lir::ChebyshevI::BandShelfBase	33
lir::Butterworth::BandShelfBase	33
lir::ChebyshevII::BandShelfBase	34
lir::ChebyshevII::BandStop< MaxOrder, StateType >	34
lir::RBJ::BandStop	35
lir::ChebyshevI::BandStop< MaxOrder, StateType >	36
lir::Bessel::BandStop< MaxOrder, StateType >	37
lir::Butterworth::BandStop< MaxOrder, StateType >	38
lir::Bessel::BandStopBase	39
lir::ChebyshevI::BandStopBase	39
lir::Butterworth::BandStopBase	40
lir::ChebyshevII::BandStopBase	40
lir::BandStopTransform	40
lir::Biquad	41
lir::BiquadBase	41
lir::BiquadPoleState	42
lir::Cascade	42
lir::CascadeStages< MaxStages, StateType >	43
lir::ComplexPair	43
lir::DirectFormI	43
lir::DirectFormII	44
lir::EnvelopeFollower< Channels, Value >	44

lir::ChebyshevII::HighPass< MaxOrder, StateType >	44
lir::Butterworth::HighPass< MaxOrder, StateType >	45
lir::RBJ::HighPass	46
lir::Bessel::HighPass< MaxOrder, StateType >	47
lir::ChebyshevI::HighPass< MaxOrder, StateType >	48
lir::ChebyshevI::HighPassBase	49
lir::Butterworth::HighPassBase	50
lir::Bessel::HighPassBase	50
lir::ChebyshevII::HighPassBase	50
lir::HighPassTransform	51
lir::Butterworth::HighShelf< MaxOrder, StateType >	51
lir::ChebyshevII::HighShelf< MaxOrder, StateType >	52
lir::RBJ::HighShelf	53
lir::ChebyshevI::HighShelf< MaxOrder, StateType >	54
lir::ChebyshevI::HighShelfBase	55
lir::Butterworth::HighShelfBase	55
lir::ChebyshevII::HighShelfBase	55
lir::RBJ::IIRNotch	56
lir::Layout< MaxPoles >	57
lir::LayoutBase	57
lir::ChebyshevI::LowPass< MaxOrder, StateType >	58
lir::Bessel::LowPass< MaxOrder, StateType >	59
lir::RBJ::LowPass	60
lir::Butterworth::LowPass< MaxOrder, StateType >	61
lir::ChebyshevII::LowPass< MaxOrder, StateType >	62
lir::Bessel::LowPassBase	63
lir::Butterworth::LowPassBase	63
lir::ChebyshevI::LowPassBase	64
lir::ChebyshevII::LowPassBase	64
lir::LowPassTransform	64
lir::Butterworth::LowShelf< MaxOrder, StateType >	65
lir::ChebyshevII::LowShelf< MaxOrder, StateType >	66

lir::RBJ::LowShelf	67
lir::ChebyshevI::LowShelf< MaxOrder, StateType >	67
lir::Bessel::LowShelfBase	68
lir::ChebyshevI::LowShelfBase	69
lir::Butterworth::LowShelfBase	69
lir::ChebyshevII::LowShelfBase	69
lir::Custom::OnePole	70
lir::PoleFilter< BaseClass, StateType, MaxAnalogPoles, MaxDigitalPoles >	70
lir::PoleFilterBase< AnalogPrototype >	71
lir::PoleFilterBase2	72
lir::PoleZeroPair	72
lir::RBJ::RBJbase	73
lir::RootFinder< maxdegree >	74
lir::RootFinderBase	74
lir::SlopeDetector< Channels, Value >	75
lir::Cascade::Stage	75
lir::BiquadBase::State< StateType >	75
lir::Cascade::Storage	75
lir::TransposedDirectFormII	76
lir::Custom::TwoPole	76
lir::Bessel::Workspace< MaxOrder >	76
lir::Bessel::WorkspaceBase	76

5 Namespace Documentation

5.1 lir Namespace Reference

Namespaces

- [Bessel](#)
- [Butterworth](#)
- [ChebyshevI](#)
- [ChebyshevII](#)
- [RBJ](#)

Classes

- class [BandPassTransform](#)
- class [BandStopTransform](#)
- class [Biquad](#)
- class [BiquadBase](#)
- struct [BiquadPoleState](#)
- class [Cascade](#)
- class [CascadeStages](#)
- struct [ComplexPair](#)
- class [DirectFormI](#)
- class [DirectFormII](#)
- class [EnvelopeFollower](#)
- class [HighPassTransform](#)
- class [Layout](#)
- class [LayoutBase](#)
- class [LowPassTransform](#)
- struct [PoleFilter](#)
- class [PoleFilterBase](#)
- class [PoleFilterBase2](#)
- struct [PoleZeroPair](#)
- struct [RootFinder](#)
- class [RootFinderBase](#)
- class [SlopeDetector](#)
- class [TransposedDirectFormII](#)

Enumerations

- enum [Kind](#)

Functions

- `template<class Td , class Ts >`
void [add](#) (int samples, Td *dest, Ts const *src, int destSkip=0, int srcSkip=0)
- `template<typename Td , typename Ts >`
void [add](#) (int channels, int samples, Td *const *dest, Ts const *const *src)
- `template<typename Td , typename Ts >`
void [copy](#) (int samples, Td *dest, Ts const *src, int destSkip=0, int srcSkip=0)

5.1.1 Detailed Description

"A Collection of Useful C++ Classes for Digital Signal Processing" By Vinnie Falco and Bernd Porr

Official project location: <https://github.com/berndporr/iir1>

See Documentation.cpp for contact information, notes, and bibliography.

License: MIT License (<http://www.opensource.org/licenses/mit-license.php>) Copyright (c) 2009 by Vinnie Falco Copyright (c) 2011 by Bernd Porr

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights

to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

"A Collection of Useful C++ Classes for Digital Signal Processing" By Vinnie Falco and Bernd Porr

Official project location: <https://github.com/berndporr/iir1>

See Documentation.cpp for contact information, notes, and bibliography.

License: MIT License (<http://www.opensource.org/licenses/mit-license.php>) Copyright (c) 2009 by Vinnie Falco Copyright (c) 2011 by Bernd Porr

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE. Describes a filter as a collection of poles and zeros along with normalization information to achieve a specified gain at a specified frequency. The poles and zeros may lie either in the s or the z plane.

5.1.2 Enumeration Type Documentation

5.1.2.1 Kind

```
enum Iir::Kind
```

Identifies the general class of filter

5.1.3 Function Documentation

5.1.3.1 add() [1/2]

```
template<class Td , class Ts >
void Iir::add (
    int samples,
    Td * dest,
    Ts const * src,
    int destSkip = 0,
    int srcSkip = 0 )
```

Utilities

These routines are handy for manipulating buffers of samples. Add src samples to dest, without clip or overflow checking.

5.1.3.2 add() [2/2]

```
template<typename Td , typename Ts >
void Iir::add (
    int channels,
    int samples,
    Td *const * dest,
    Ts const *const * src )
```

Multichannel add

5.1.3.3 copy()

```
template<typename Td , typename Ts >
void Iir::copy (
    int samples,
    Td * dest,
    Ts const * src,
    int destSkip = 0,
    int srcSkip = 0 )
```

Copy samples from src to dest, which may not overlap. Performs an implicit type conversion if Ts and Td are different (for example, float to double).

5.2 Iir::Bessel Namespace Reference

Classes

- class [AnalogLowPass](#)
- class [AnalogLowShelf](#)
- struct [BandPass](#)
- struct [BandPassBase](#)
- struct [BandStop](#)
- struct [BandStopBase](#)
- struct [HighPass](#)
- struct [HighPassBase](#)
- struct [LowPass](#)
- struct [LowPassBase](#)
- struct [LowShelfBase](#)
- struct [Workspace](#)
- struct [WorkspaceBase](#)

5.2.1 Detailed Description

Filters with [Bessel](#) response characteristics

5.3 `lir::Butterworth` Namespace Reference

Classes

- class [AnalogLowPass](#)
- class [AnalogLowShelf](#)
- struct [BandPass](#)
- struct [BandPassBase](#)
- struct [BandShelf](#)
- struct [BandShelfBase](#)
- struct [BandStop](#)
- struct [BandStopBase](#)
- struct [HighPass](#)
- struct [HighPassBase](#)
- struct [HighShelf](#)
- struct [HighShelfBase](#)
- struct [LowPass](#)
- struct [LowPassBase](#)
- struct [LowShelf](#)
- struct [LowShelfBase](#)

5.3.1 Detailed Description

Filters with [Butterworth](#) response characteristics

5.4 `lir::ChebyshevI` Namespace Reference

Classes

- class [AnalogLowPass](#)
- class [AnalogLowShelf](#)
- struct [BandPass](#)
- struct [BandPassBase](#)
- struct [BandShelf](#)
- struct [BandShelfBase](#)
- struct [BandStop](#)
- struct [BandStopBase](#)
- struct [HighPass](#)
- struct [HighPassBase](#)
- struct [HighShelf](#)
- struct [HighShelfBase](#)
- struct [LowPass](#)
- struct [LowPassBase](#)
- struct [LowShelf](#)
- struct [LowShelfBase](#)

5.4.1 Detailed Description

Filters with Chebyshev response characteristics. The last parameter defines the passband ripple in decibel.

5.5 `lir::ChebyshevII` Namespace Reference

Classes

- class [AnalogLowPass](#)
- class [AnalogLowShelf](#)
- struct [BandPass](#)
- struct [BandPassBase](#)
- struct [BandShelf](#)
- struct [BandShelfBase](#)
- struct [BandStop](#)
- struct [BandStopBase](#)
- struct [HighPass](#)
- struct [HighPassBase](#)
- struct [HighShelf](#)
- struct [HighShelfBase](#)
- struct [LowPass](#)
- struct [LowPassBase](#)
- struct [LowShelf](#)
- struct [LowShelfBase](#)

5.5.1 Detailed Description

Filters with [ChebyshevII](#) response characteristics. The last parameter defines the minimal stopband rejection requested. Generally there will be frequencies where the rejection is much better but this parameter guarantees that the rejection is at least as specified.

5.6 `lir::RBJ` Namespace Reference

Classes

- struct [AllPass](#)
- struct [BandPass1](#)
- struct [BandPass2](#)
- struct [BandShelf](#)
- struct [BandStop](#)
- struct [HighPass](#)
- struct [HighShelf](#)
- struct [IIRNotch](#)
- struct [LowPass](#)
- struct [LowShelf](#)
- struct [RBJbase](#)

5.6.1 Detailed Description

Filter realizations based on Robert Bristol-Johnson formulae:

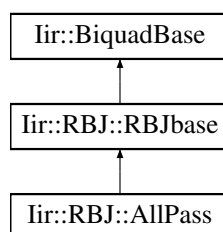
<http://www.musicdsp.org/files/Audio-EQ-Cookbook.txt>

These are all 2nd order filters which are tuned with the Q (or Quality factor). The Q factor causes a resonance at the cutoff frequency. The higher the Q factor the higher the resonance. If $0.5 < Q < 1/\sqrt{2}$ then there is no resonance peak. Above $1/\sqrt{2}$ the peak becomes more and more pronounced. For bandpass and stopband the Q factor is replaced by the width of the filter. The higher Q the more narrow the bandwidth of the notch or bandpass.

6 Class Documentation

6.1 Iir::RBJ::AllPass Struct Reference

Inheritance diagram for Iir::RBJ::AllPass:



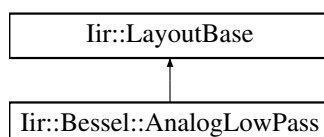
Additional Inherited Members

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

- iir/RBJ.h
- iir/RBJ.cpp

6.2 Iir::Bessel::AnalogLowPass Class Reference

Inheritance diagram for Iir::Bessel::AnalogLowPass:

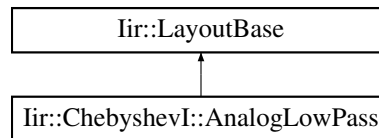


The documentation for this class was generated from the following files:

- iir/Bessel.h
- iir/Bessel.cpp

6.3 Iir::ChebyshevI::AnalogLowPass Class Reference

Inheritance diagram for Iir::ChebyshevI::AnalogLowPass:



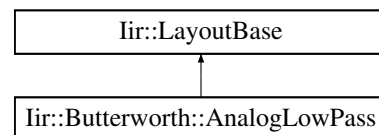
The documentation for this class was generated from the following files:

- iir/ChebyshevI.h
- iir/ChebyshevI.cpp

6.4 Iir::Butterworth::AnalogLowPass Class Reference

```
#include <Butterworth.h>
```

Inheritance diagram for Iir::Butterworth::AnalogLowPass:



6.4.1 Detailed Description

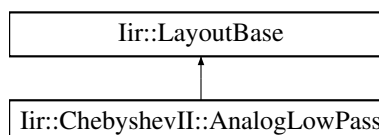
Half-band analog prototypes (s-plane)

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

- iir/Butterworth.h
- iir/Butterworth.cpp

6.5 Iir::ChebyshevII::AnalogLowPass Class Reference

Inheritance diagram for Iir::ChebyshevII::AnalogLowPass:

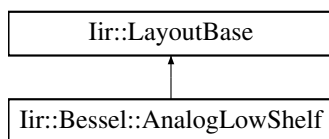


The documentation for this class was generated from the following files:

- iir/ChebyshevII.h
- iir/ChebyshevII.cpp

6.6 Iir::Bessel::AnalogLowShelf Class Reference

Inheritance diagram for Iir::Bessel::AnalogLowShelf:

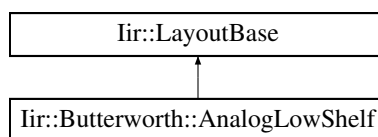


The documentation for this class was generated from the following files:

- iir/Bessel.h
- iir/Bessel.cpp

6.7 Iir::Butterworth::AnalogLowShelf Class Reference

Inheritance diagram for Iir::Butterworth::AnalogLowShelf:

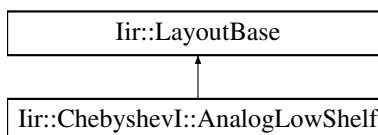


The documentation for this class was generated from the following files:

- iir/Butterworth.h
- iir/Butterworth.cpp

6.8 Iir::ChebyshevI::AnalogLowShelf Class Reference

Inheritance diagram for Iir::ChebyshevI::AnalogLowShelf:

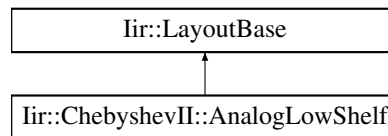


The documentation for this class was generated from the following files:

- iir/ChebyshevI.h
- iir/ChebyshevI.cpp

6.9 Iir::ChebyshevII::AnalogLowShelf Class Reference

Inheritance diagram for Iir::ChebyshevII::AnalogLowShelf:



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

- iir/ChebyshevII.h
- iir/ChebyshevII.cpp

6.10 Iir::RootFinderBase::Array Struct Reference

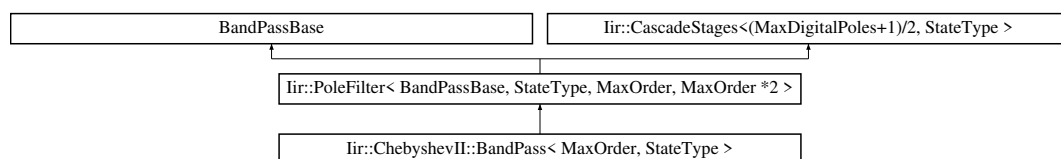
The documentation for this struct was generated from the following file:

- iir/RootFinder.h

6.11 Iir::ChebyshevII::BandPass< MaxOrder, StateType > Struct Template Reference

```
#include <ChebyshevII.h>
```

Inheritance diagram for Iir::ChebyshevII::BandPass< MaxOrder, StateType >:



Public Member Functions

- void [setup](#) (double sampleRate, double centerFrequency, double widthFrequency, double stopBandDb)

6.11.1 Detailed Description

```
template<int MaxOrder, class StateType = DEFAULT_STATE>
struct Iir::ChebyshevII::BandPass< MaxOrder, StateType >
```

[ChebyshevII](#) bandpass filter

6.11.2 Member Function Documentation

6.11.2.1 setup()

```
template<int MaxOrder, class StateType = DEFAULT_STATE>
void Iir::ChebyshevII::BandPass< MaxOrder, StateType >::setup (
    double sampleRate,
    double centerFrequency,
    double widthFrequency,
    double stopBandDb ) [inline]
```

Calculates the coefficients of the filter

Parameters

<i>sampleRate</i>	Sampling rate
<i>centerFrequency</i>	Center frequency of the bandpass
<i>widthFrequency</i>	Width of the bandpass
<i>stopBandDb</i>	Permitted ripples in dB in the stopband

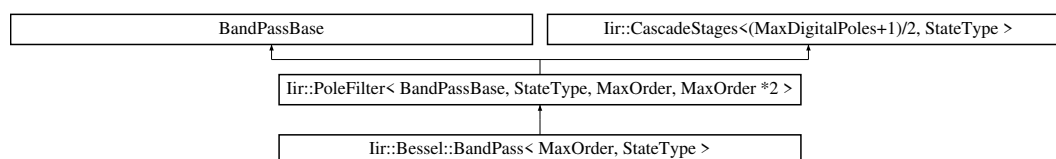
The documentation for this struct was generated from the following file:

- iir/ChebyshevII.h

6.12 Iir::Bessel::BandPass< MaxOrder, StateType > Struct Template Reference

```
#include <Bessel.h>
```

Inheritance diagram for Iir::Bessel::BandPass< MaxOrder, StateType >:



Public Member Functions

- void `setup` (double sampleRate, double centerFrequency, double widthFrequency)

6.12.1 Detailed Description

```
template<int MaxOrder, class StateType = DEFAULT_STATE>
struct Iir::Bessel::BandPass< MaxOrder, StateType >
```

[Bessel](#) bandpass.

6.12.2 Member Function Documentation

6.12.2.1 setup()

```
template<int MaxOrder, class StateType = DEFAULT_STATE>
void Iir::Bessel::BandPass< MaxOrder, StateType >::setup (
    double sampleRate,
    double centerFrequency,
    double widthFrequency ) [inline]
```

Calculate the coefficients

Parameters

<i>sampleRate</i>	Sampling rate
<i>centerFrequency</i>	Center frequency of the bandpass in Hz
<i>widthFrequency</i>	Width of the bandpass in Hz

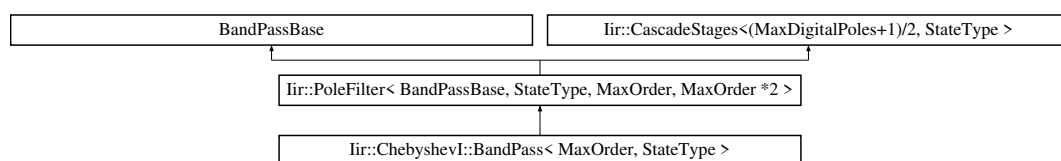
The documentation for this struct was generated from the following file:

- iir/Bessel.h

6.13 Iir::ChebyshevI::BandPass< MaxOrder, StateType > Struct Template Reference

```
#include <ChebyshevI.h>
```

Inheritance diagram for Iir::ChebyshevI::BandPass< MaxOrder, StateType >:



Public Member Functions

- void [setup](#) (double sampleRate, double centerFrequency, double widthFrequency, double rippleDb)

6.13.1 Detailed Description

```
template<int MaxOrder, class StateType = DEFAULT_STATE>
struct Iir::ChebyshevI::BandPass< MaxOrder, StateType >
```

[ChebyshevI](#) bandpass filter

6.13.2 Member Function Documentation

6.13.2.1 setup()

```
template<int MaxOrder, class StateType = DEFAULT_STATE>
void Iir::ChebyshevI::BandPass< MaxOrder, StateType >::setup (
    double sampleRate,
    double centerFrequency,
    double widthFrequency,
    double rippleDb ) [inline]
```

Calculates the coefficients of the filter

Parameters

<i>sampleRate</i>	Sampling rate
<i>centerFrequency</i>	Center frequency of the bandpass
<i>widthFrequency</i>	Frequency with of the passband
<i>rippleDb</i>	Permitted ripples in dB in the passband

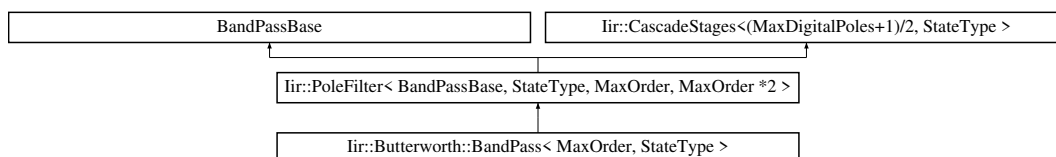
The documentation for this struct was generated from the following file:

- iir/ChebyshevI.h

6.14 Iir::Butterworth::BandPass< MaxOrder, StateType > Struct Template Reference

```
#include <Butterworth.h>
```

Inheritance diagram for Iir::Butterworth::BandPass< MaxOrder, StateType >:



Public Member Functions

- void `setup` (double sampleRate, double centerFrequency, double widthFrequency)

6.14.1 Detailed Description

```
template<int MaxOrder, class StateType = DEFAULT_STATE>
struct Iir::Butterworth::BandPass< MaxOrder, StateType >
```

Butterworth Bandpass filter.

6.14.2 Member Function Documentation

6.14.2.1 setup()

```
template<int MaxOrder, class StateType = DEFAULT_STATE>
void Iir::Butterworth::BandPass< MaxOrder, StateType >::setup (
    double sampleRate,
    double centerFrequency,
    double widthFrequency ) [inline]
```

Calculates the coefficients

Parameters

<i>sampleRate</i>	Sampling rate
<i>centerFrequency</i>	Centre frequency of the bandpass
<i>widthFrequency</i>	Width of the bandpass

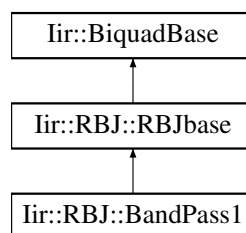
The documentation for this struct was generated from the following file:

- iir/Butterworth.h

6.15 Iir::RBJ::BandPass1 Struct Reference

```
#include <RBJ.h>
```

Inheritance diagram for Iir::RBJ::BandPass1:



Public Member Functions

- void [setup](#) (double sampleRate, double centerFrequency, double bandWidth)

6.15.1 Detailed Description

Bandpass with constant skirt gain

6.15.2 Member Function Documentation

6.15.2.1 setup()

```
void Iir::RBJ::BandPass1::setup (
    double sampleRate,
    double centerFrequency,
    double bandWidth )
```

Calculates the coefficients

Parameters

<i>sampleRate</i>	Sampling rate
<i>centerFrequency</i>	Center frequency of the bandpass
<i>bandWidth</i>	Bandwidth of the bandpass

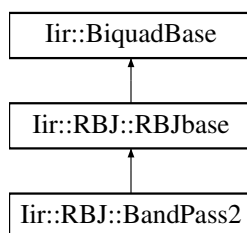
The documentation for this struct was generated from the following files:

- iir/RBJ.h
- iir/RBJ.cpp

6.16 Iir::RBJ::BandPass2 Struct Reference

```
#include <RBJ.h>
```

Inheritance diagram for Iir::RBJ::BandPass2:



Public Member Functions

- void [setup](#) (double *sampleRate*, double *centerFrequency*, double *bandWidth*)

6.16.1 Detailed Description

Bandpass with constant 0 dB peak gain

6.16.2 Member Function Documentation

6.16.2.1 setup()

```
void Iir::RBJ::BandPass2::setup (
    double sampleRate,
    double centerFrequency,
    double bandWidth )
```

Calculates the coefficients

Parameters

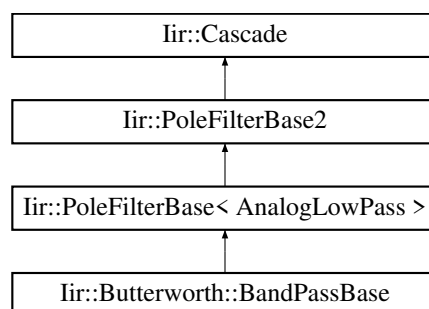
<i>sampleRate</i>	Sampling rate
<i>centerFrequency</i>	Center frequency of the bandpass
<i>bandWidth</i>	Bandwidth of the bandpass

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

- iir/RBJ.h
- iir/RBJ.cpp

6.17 Iir::Butterworth::BandPassBase Struct Reference

Inheritance diagram for Iir::Butterworth::BandPassBase:

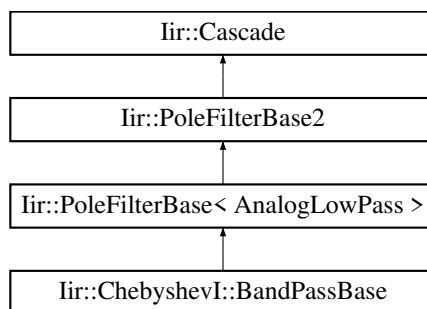


The documentation for this struct was generated from the following files:

- iir/Butterworth.h
- iir/Butterworth.cpp

6.18 Iir::ChebyshevI::BandPassBase Struct Reference

Inheritance diagram for Iir::ChebyshevI::BandPassBase:

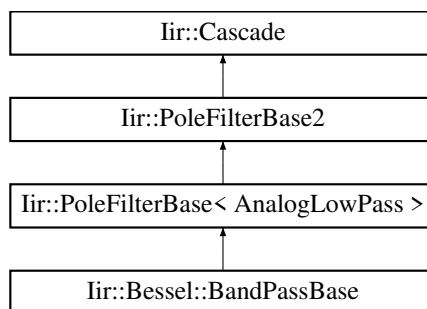


The documentation for this struct was generated from the following files:

- iir/ChebyshevI.h
- iir/ChebyshevI.cpp

6.19 Iir::Bessel::BandPassBase Struct Reference

Inheritance diagram for Iir::Bessel::BandPassBase:

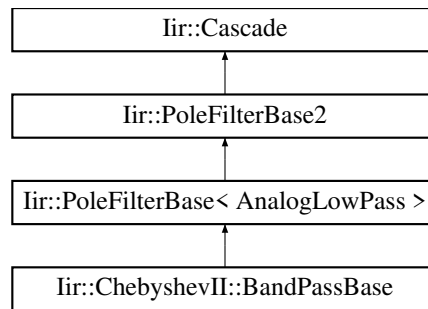


The documentation for this struct was generated from the following files:

- iir/Bessel.h
- iir/Bessel.cpp

6.20 Iir::ChebyshevII::BandPassBase Struct Reference

Inheritance diagram for Iir::ChebyshevII::BandPassBase:



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

- iir/ChebyshevII.h
- iir/ChebyshevII.cpp

6.21 Iir::BandPassTransform Class Reference

```
#include <PoleFilter.h>
```

6.21.1 Detailed Description

low pass to band pass transform

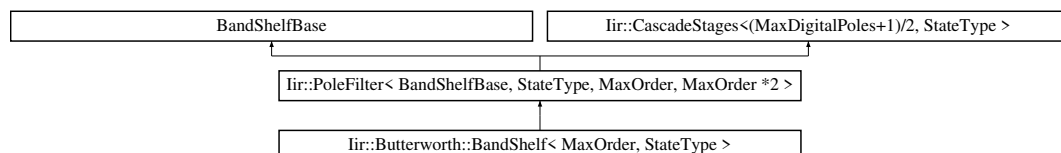
The documentation for this class was generated from the following files:

- iir/PoleFilter.h
- iir/PoleFilter.cpp

6.22 Iir::Butterworth::BandShelf< MaxOrder, StateType > Struct Template Reference

```
#include <Butterworth.h>
```

Inheritance diagram for `Iir::Butterworth::BandShelf< MaxOrder, StateType >`:



Public Member Functions

- void [setup](#) (double sampleRate, double centerFrequency, double widthFrequency, double gainDb)

6.22.1 Detailed Description

```
template<int MaxOrder, class StateType = DEFAULT_STATE>
struct Iir::Butterworth::BandShelf< MaxOrder, StateType >
```

Butterworth Bandshef filter: it is a bandpass filter which amplifies at a specified gain in dB the frequencies in the passband.

6.22.2 Member Function Documentation

6.22.2.1 setup()

```
template<int MaxOrder, class StateType = DEFAULT_STATE>
void Iir::Butterworth::BandShelf< MaxOrder, StateType >::setup (
    double sampleRate,
    double centerFrequency,
    double widthFrequency,
    double gainDb ) [inline]
```

Calculates the coefficients

Parameters

<i>sampleRate</i>	Sampling rate
<i>centerFrequency</i>	Centre frequency of the passband
<i>widthFrequency</i>	Width of the passband
<i>gainDb</i>	The gain in the passband

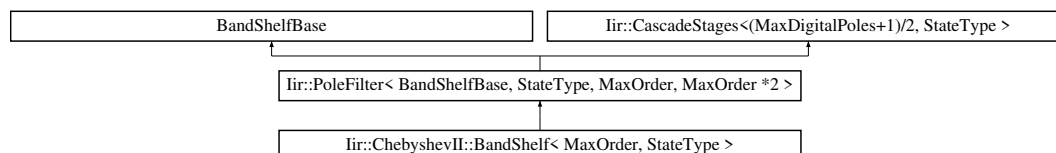
The documentation for this struct was generated from the following file:

- iir/Butterworth.h

6.23 Iir::ChebyshevII::BandShelf< MaxOrder, StateType > Struct Template Reference

```
#include <ChebyshevII.h>
```

Inheritance diagram for Iir::ChebyshevII::BandShelf< MaxOrder, StateType >:



Public Member Functions

- void **setup** (double sampleRate, double centerFrequency, double widthFrequency, double gainDb, double stopBandDb)

6.23.1 Detailed Description

```
template<int MaxOrder, class StateType = DEFAULT_STATE>
struct Iir::ChebyshevII::BandShelf< MaxOrder, StateType >
```

[ChebyshevII](#) bandshelf filter. Bandpass with specified gain and 0 dB gain in the stopband.

6.23.2 Member Function Documentation

6.23.2.1 setup()

```
template<int MaxOrder, class StateType = DEFAULT_STATE>
void Iir::ChebyshevII::BandShelf< MaxOrder, StateType >::setup (
    double sampleRate,
    double centerFrequency,
    double widthFrequency,
    double gainDb,
    double stopBandDb ) [inline]
```

Calculates the coefficients of the filter

Parameters

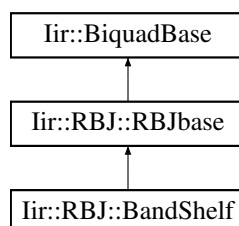
<i>sampleRate</i>	Sampling rate
<i>centerFrequency</i>	Center frequency of the bandpass
<i>widthFrequency</i>	Width of the bandpass
<i>gainDb</i>	Gain in the passband. The stopband has always 0dB.
<i>stopBandDb</i>	Permitted ripples in dB in the stopband

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

- iir/ChebyshevII.h

6.24 Iir::RBJ::BandShelf Struct Reference

Inheritance diagram for Iir::RBJ::BandShelf:



Additional Inherited Members

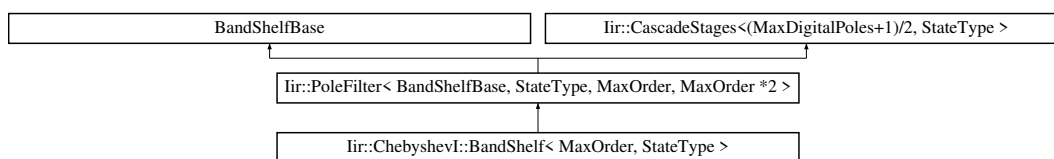
The documentation for this struct was generated from the following files:

- iir/RBJ.h
- iir/RBJ.cpp

6.25 Iir::ChebyshevI::BandShelf< MaxOrder, StateType > Struct Template Reference

```
#include <ChebyshevI.h>
```

Inheritance diagram for Iir::ChebyshevI::BandShelf< MaxOrder, StateType >:



Public Member Functions

- void [setup](#) (double sampleRate, double centerFrequency, double widthFrequency, double gainDb, double rippleDb)

6.25.1 Detailed Description

```
template<int MaxOrder, class StateType = DEFAULT_STATE>
struct Iir::ChebyshevI::BandShelf< MaxOrder, StateType >
```

[ChebyshevI](#) bandshelf filter. Specified gain in the passband. Otherwise 0 dB.

6.25.2 Member Function Documentation

6.25.2.1 setup()

```
template<int MaxOrder, class StateType = DEFAULT_STATE>
void Iir::ChebyshevI::BandShelf< MaxOrder, StateType >::setup (
    double sampleRate,
    double centerFrequency,
    double widthFrequency,
    double gainDb,
    double rippleDb ) [inline]
```

Calculates the coefficients of the filter

Parameters

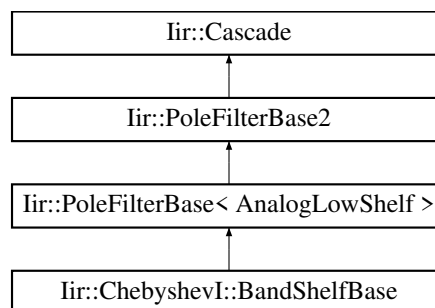
<i>sampleRate</i>	Sampling rate
<i>centerFrequency</i>	Center frequency of the passband
<i>widthFrequency</i>	Width of the passband.
<i>gainDb</i>	Gain in the passband. The stopband has 0 dB.
<i>rippleDb</i>	Permitted ripples in dB in the passband.

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

- iir/ChebyshevI.h

6.26 Iir::ChebyshevI::BandShelfBase Struct Reference

Inheritance diagram for Iir::ChebyshevI::BandShelfBase:

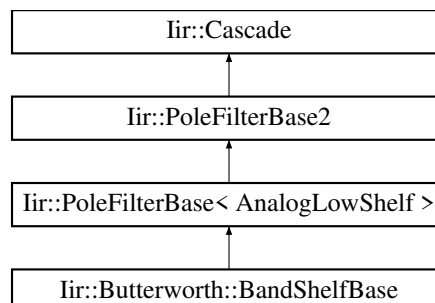


The documentation for this struct was generated from the following files:

- iir/ChebyshevI.h
- iir/ChebyshevI.cpp

6.27 Iir::Butterworth::BandShelfBase Struct Reference

Inheritance diagram for Iir::Butterworth::BandShelfBase:

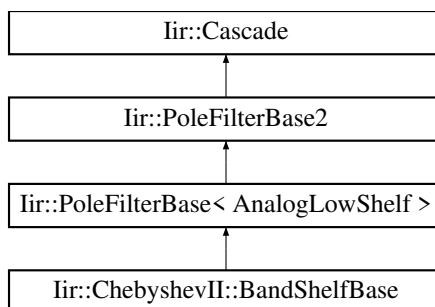


The documentation for this struct was generated from the following files:

- iir/Butterworth.h
- iir/Butterworth.cpp

6.28 Iir::ChebyshevII::BandShelfBase Struct Reference

Inheritance diagram for Iir::ChebyshevII::BandShelfBase:



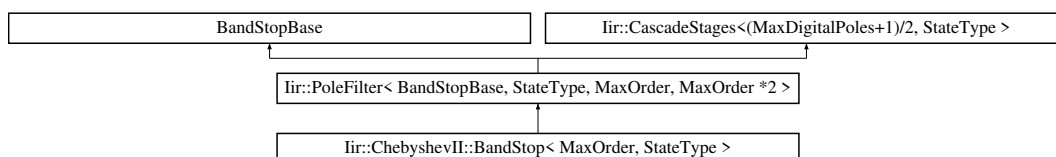
The documentation for this struct was generated from the following files:

- iir/ChebyshevII.h
- iir/ChebyshevII.cpp

6.29 Iir::ChebyshevII::BandStop< MaxOrder, StateType > Struct Template Reference

```
#include <ChebyshevII.h>
```

Inheritance diagram for Iir::ChebyshevII::BandStop< MaxOrder, StateType >:



Public Member Functions

- void [setup](#) (double sampleRate, double centerFrequency, double widthFrequency, double stopBandDb)

6.29.1 Detailed Description

```
template<int MaxOrder, class StateType = DEFAULT_STATE>
struct Iir::ChebyshevII::BandStop< MaxOrder, StateType >
```

[ChebyshevII](#) bandstop filter.

6.29.2 Member Function Documentation

6.29.2.1 setup()

```
template<int MaxOrder, class StateType = DEFAULT_STATE>
void Iir::ChebyshevII::BandStop< MaxOrder, StateType >::setup (
    double sampleRate,
    double centerFrequency,
    double widthFrequency,
    double stopBandDb ) [inline]
```

Calculates the coefficients of the filter

Parameters

<i>sampleRate</i>	Sampling rate
<i>centerFrequency</i>	Center frequency of the bandstop
<i>widthFrequency</i>	Width of the bandstop
<i>stopBandDb</i>	Permitted ripples in dB in the stopband

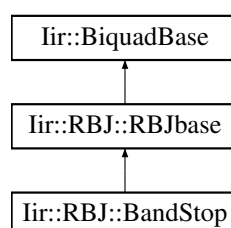
The documentation for this struct was generated from the following file:

- iir/ChebyshevII.h

6.30 Iir::RBJ::BandStop Struct Reference

```
#include <RBJ.h>
```

Inheritance diagram for Iir::RBJ::BandStop:



Public Member Functions

- void [setup](#) (double sampleRate, double centerFrequency, double bandWidth)

6.30.1 Detailed Description

Bandstop filter. Warning: the bandwidth might not be accurate for narrow notches.

6.30.2 Member Function Documentation

6.30.2.1 setup()

```
void Iir::RBJ::BandStop::setup (
    double sampleRate,
    double centerFrequency,
    double bandWidth )
```

Calculates the coefficients

Parameters

<i>sampleRate</i>	Sampling rate
<i>centerFrequency</i>	Center frequency of the bandstop
<i>bandWidth</i>	Bandwidth of the bandstop

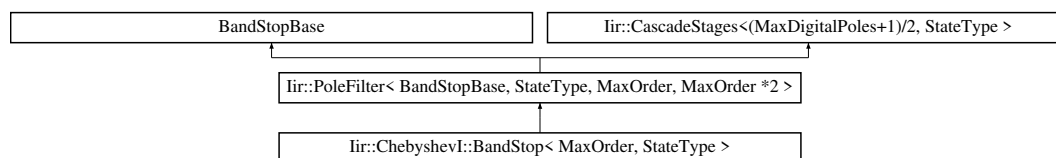
The documentation for this struct was generated from the following files:

- iir/RBJ.h
- iir/RBJ.cpp

6.31 Iir::ChebyshevI::BandStop< MaxOrder, StateType > Struct Template Reference

```
#include <ChebyshevI.h>
```

Inheritance diagram for Iir::ChebyshevI::BandStop< MaxOrder, StateType >:



Public Member Functions

- void [setup](#) (double sampleRate, double centerFrequency, double widthFrequency, double rippleDb)

6.31.1 Detailed Description

```
template<int MaxOrder, class StateType = DEFAULT_STATE>
struct Iir::ChebyshevI::BandStop< MaxOrder, StateType >
```

[ChebyshevI](#) bandstop filter

6.31.2 Member Function Documentation

6.31.2.1 setup()

```
template<int MaxOrder, class StateType = DEFAULT_STATE>
void Iir::ChebyshevI::BandStop< MaxOrder, StateType >::setup (
    double sampleRate,
    double centerFrequency,
    double widthFrequency,
    double rippleDb ) [inline]
```

Calculates the coefficients of the filter

Parameters

<i>sampleRate</i>	Sampling rate
<i>centerFrequency</i>	Center frequency of the notch
<i>widthFrequency</i>	Frequency with of the notch
<i>rippleDb</i>	Permitted ripples in dB in the passband

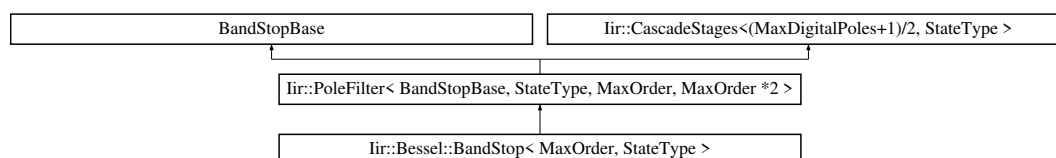
The documentation for this struct was generated from the following file:

- iir/ChebyshevI.h

6.32 Iir::Bessel::BandStop< MaxOrder, StateType > Struct Template Reference

```
#include <Bessel.h>
```

Inheritance diagram for Iir::Bessel::BandStop< MaxOrder, StateType >:



Public Member Functions

- void [setup](#) (double sampleRate, double centerFrequency, double widthFrequency)

6.32.1 Detailed Description

```
template<int MaxOrder, class StateType = DEFAULT_STATE>
struct Iir::Bessel::BandStop< MaxOrder, StateType >
```

[Bessel](#) bandstop.

6.32.2 Member Function Documentation

6.32.2.1 setup()

```
template<int MaxOrder, class StateType = DEFAULT_STATE>
void Iir::Bessel::BandStop< MaxOrder, StateType >::setup (
    double sampleRate,
    double centerFrequency,
    double widthFrequency ) [inline]
```

Calculate the coefficients

Parameters

<i>sampleRate</i>	Sampling rate
<i>centerFrequency</i>	Center frequency of the bandpass in Hz
<i>widthFrequency</i>	Width of the bandpass in Hz

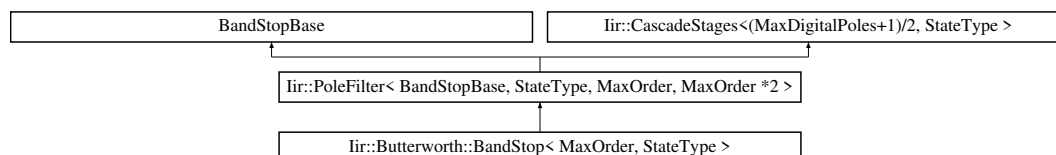
The documentation for this struct was generated from the following file:

- iir/Bessel.h

6.33 Iir::Butterworth::BandStop< MaxOrder, StateType > Struct Template Reference

```
#include <Butterworth.h>
```

Inheritance diagram for Iir::Butterworth::BandStop< MaxOrder, StateType >:



Public Member Functions

- void [setup](#) (double sampleRate, double centerFrequency, double widthFrequency)

6.33.1 Detailed Description

```
template<int MaxOrder, class StateType = DEFAULT_STATE>
struct Iir::Butterworth::BandStop< MaxOrder, StateType >
```

[Butterworth](#) Bandstop filter.

6.33.2 Member Function Documentation

6.33.2.1 setup()

```
template<int MaxOrder, class StateType = DEFAULT_STATE>
void Iir::Butterworth::BandStop< MaxOrder, StateType >::setup (
    double sampleRate,
    double centerFrequency,
    double widthFrequency ) [inline]
```

Calculates the coefficients

Parameters

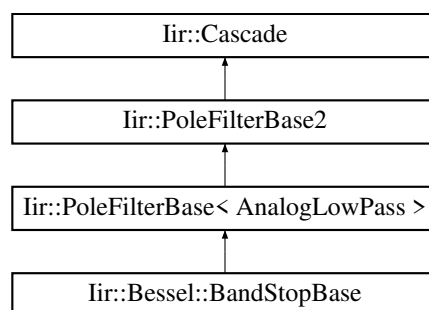
<i>sampleRate</i>	Sampling rate
<i>centerFrequency</i>	Centre frequency of the bandstop
<i>widthFrequency</i>	Width of the bandstop

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

- iir/Butterworth.h

6.34 Iir::Bessel::BandStopBase Struct Reference

Inheritance diagram for Iir::Bessel::BandStopBase:

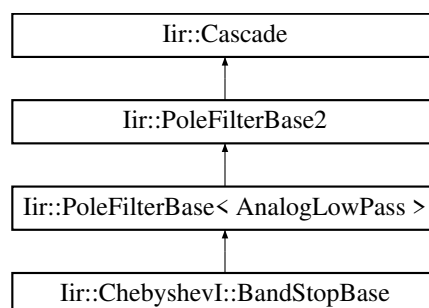


The documentation for this struct was generated from the following files:

- iir/Bessel.h
- iir/Bessel.cpp

6.35 Iir::ChebyshevI::BandStopBase Struct Reference

Inheritance diagram for Iir::ChebyshevI::BandStopBase:

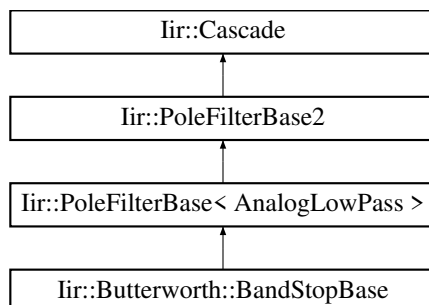


The documentation for this struct was generated from the following files:

- iir/ChebyshevI.h
- iir/ChebyshevI.cpp

6.36 Iir::Butterworth::BandStopBase Struct Reference

Inheritance diagram for Iir::Butterworth::BandStopBase:

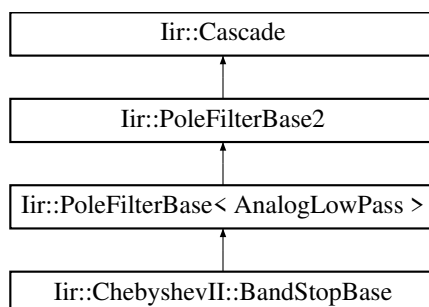


The documentation for this struct was generated from the following files:

- iir/Butterworth.h
- iir/Butterworth.cpp

6.37 Iir::ChebyshevII::BandStopBase Struct Reference

Inheritance diagram for Iir::ChebyshevII::BandStopBase:



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

- iir/ChebyshevII.h
- iir/ChebyshevII.cpp

6.38 Iir::BandStopTransform Class Reference

```
#include <PoleFilter.h>
```

6.38.1 Detailed Description

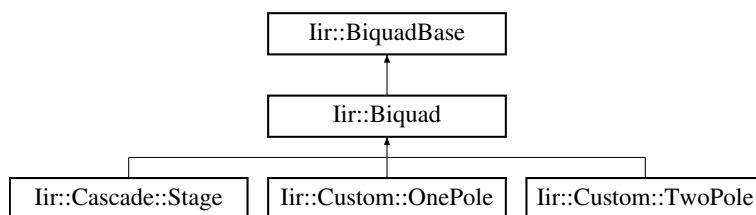
low pass to band stop transform

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

- iir/PoleFilter.h
- iir/PoleFilter.cpp

6.39 Iir::Biquad Class Reference

Inheritance diagram for Iir::Biquad:

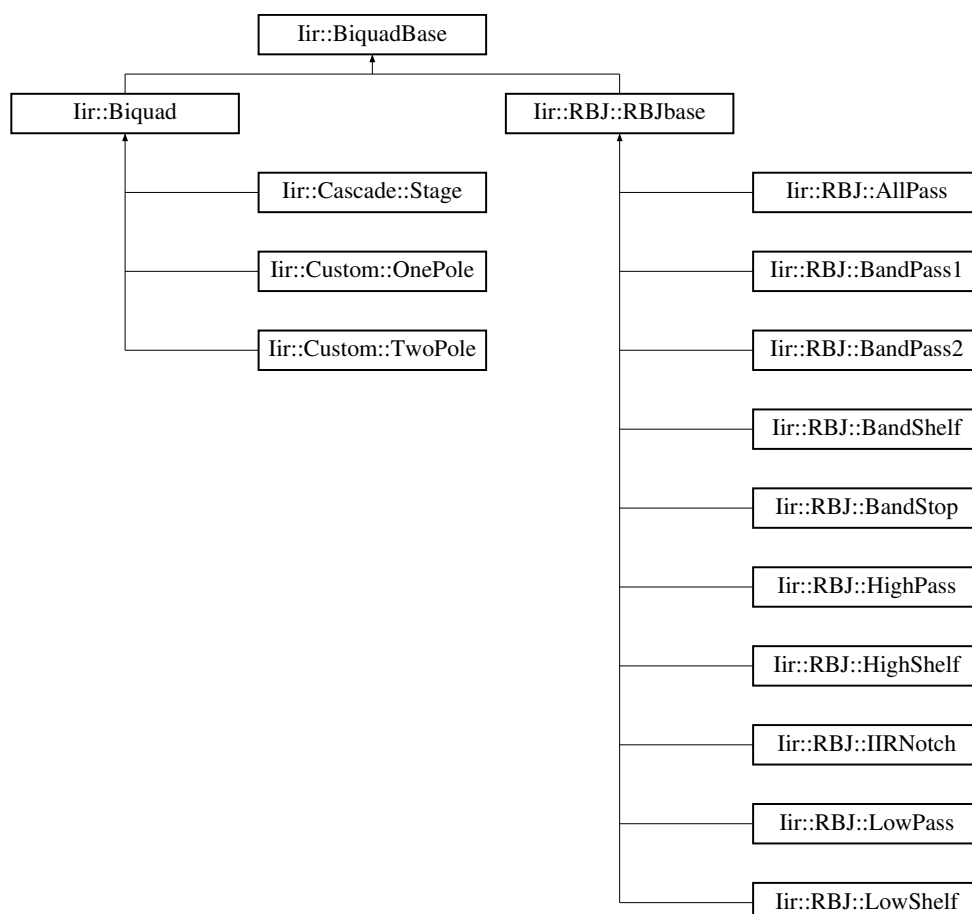


The documentation for this class was generated from the following files:

- iir/Biquad.h
- iir/Biquad.cpp

6.40 Iir::BiquadBase Class Reference

Inheritance diagram for Iir::BiquadBase:



Classes

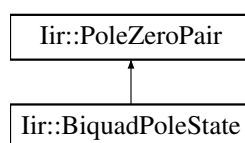
- struct [State](#)

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

- iir/Biquad.h
- iir/Biquad.cpp

6.41 Iir::BiquadPoleState Struct Reference

Inheritance diagram for Iir::BiquadPoleState:

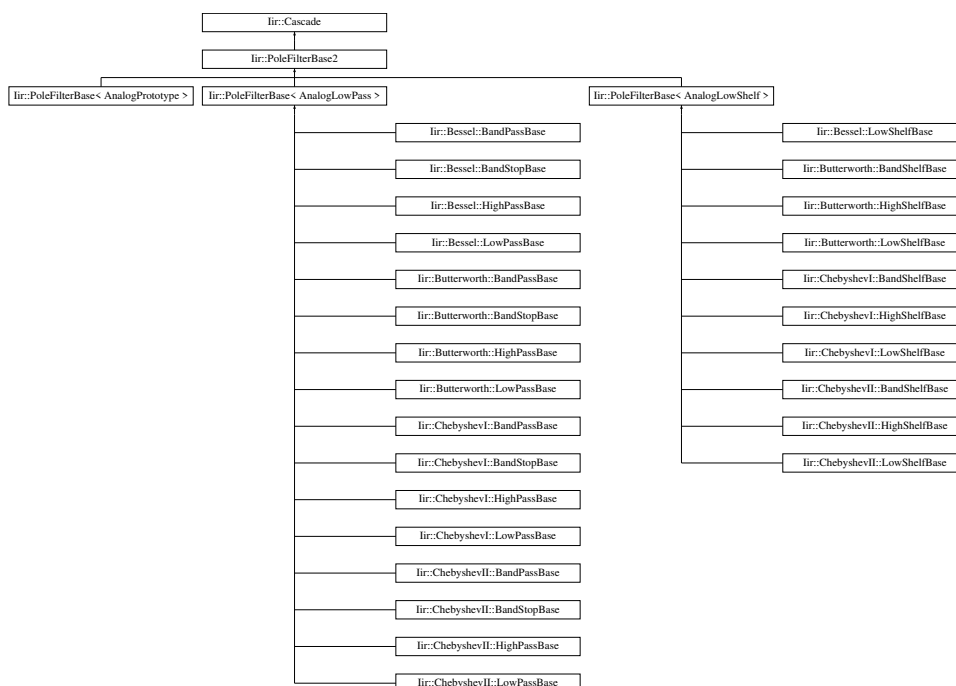


The documentation for this struct was generated from the following files:

- iir/Biquad.h
- iir/Biquad.cpp

6.42 Iir::Cascade Class Reference

Inheritance diagram for Iir::Cascade:



Classes

- struct [Stage](#)
- struct [Storage](#)

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

- iir/Cascade.h
- iir/Cascade.cpp

6.43 Iir::CascadeStages< MaxStages, StateType > Class Template Reference

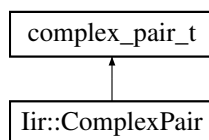
The documentation for this class was generated from the following file:

- iir/Cascade.h

6.44 Iir::ComplexPair Struct Reference

```
#include <Types.h>
```

Inheritance diagram for Iir::ComplexPair:



6.44.1 Detailed Description

A conjugate or real pair

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

- iir/Types.h

6.45 Iir::DirectFormI Class Reference

```
#include <State.h>
```

6.45.1 Detailed Description

State for applying a second order section to a sample using Direct Form I

Difference equation:

$$y[n] = (b0/a0)*x[n] + (b1/a0)*x[n-1] + (b2/a0)*x[n-2]$$

$$\bullet (a1/a0)*y[n-1] - (a2/a0)*y[n-2]$$

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

- iir/State.h

6.46 Iir::DirectFormII Class Reference

```
#include <State.h>
```

6.46.1 Detailed Description

State for applying a second order section to a sample using Direct Form II

Difference equation:

$$v[n] = x[n] - (a1/a0)*v[n-1] - (a2/a0)*v[n-2] \quad y(n) = (b0/a0)*v[n] + (b1/a0)*v[n-1] + (b2/a0)*v[n-2]$$

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

- iir/State.h

6.47 Iir::EnvelopeFollower< Channels, Value > Class Template Reference

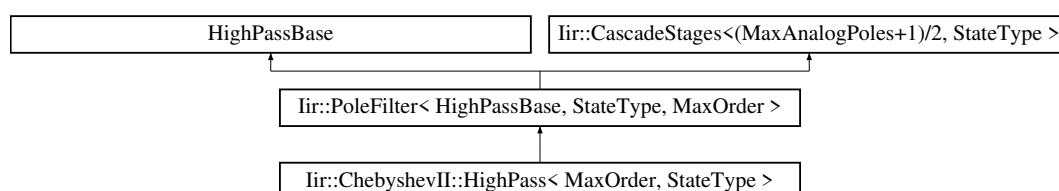
The documentation for this class was generated from the following file:

- iir/Utilities.h

6.48 Iir::ChebyshevII::HighPass< MaxOrder, StateType > Struct Template Reference

```
#include <ChebyshevII.h>
```

Inheritance diagram for Iir::ChebyshevII::HighPass< MaxOrder, StateType >:



Public Member Functions

- void [setup](#) (double sampleRate, double cutoffFrequency, double stopBandDb)

6.48.1 Detailed Description

```
template<int MaxOrder, class StateType = DEFAULT_STATE>
struct Iir::ChebyshevII::HighPass< MaxOrder, StateType >
```

[ChebyshevII](#) highpass filter

6.48.2 Member Function Documentation

6.48.2.1 setup()

```
template<int MaxOrder, class StateType = DEFAULT_STATE>
void Iir::ChebyshevII::HighPass< MaxOrder, StateType >::setup (
    double sampleRate,
    double cutoffFrequency,
    double stopBandDb ) [inline]
```

Calculates the coefficients of the filter

Parameters

<i>sampleRate</i>	Sampling rate
<i>cutoffFrequency</i>	Cutoff frequency.
<i>stopBandDb</i>	Permitted ripples in dB in the stopband

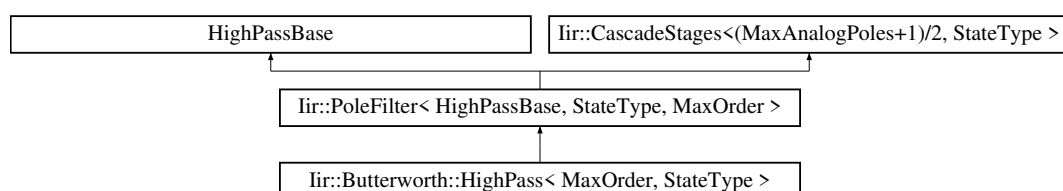
The documentation for this struct was generated from the following file:

- iir/ChebyshevII.h

6.49 Iir::Butterworth::HighPass< MaxOrder, StateType > Struct Template Reference

```
#include <Butterworth.h>
```

Inheritance diagram for Iir::Butterworth::HighPass< MaxOrder, StateType >:



Public Member Functions

- void [setup](#) (double sampleRate, double cutoffFrequency)

6.49.1 Detailed Description

```
template<int MaxOrder, class StateType = DEFAULT_STATE>
struct Iir::Butterworth::HighPass< MaxOrder, StateType >
```

[Butterworth](#) Highpass filter.

6.49.2 Member Function Documentation

6.49.2.1 setup()

```
template<int MaxOrder, class StateType = DEFAULT_STATE>
void Iir::Butterworth::HighPass< MaxOrder, StateType >::setup (
    double sampleRate,
    double cutoffFrequency ) [inline]
```

Calculates the coefficients

Parameters

<i>sampleRate</i>	Sampling rate
<i>cutoffFrequency</i>	Cutoff

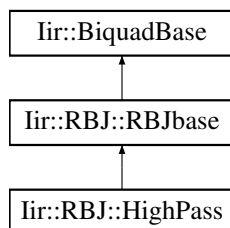
The documentation for this struct was generated from the following file:

- iir/Butterworth.h

6.50 Iir::RBJ::HighPass Struct Reference

```
#include <RBJ.h>
```

Inheritance diagram for Iir::RBJ::HighPass:



Public Member Functions

- void [setup](#) (double sampleRate, double cutoffFrequency, double q)

6.50.1 Detailed Description

Highpass.

6.50.2 Member Function Documentation

6.50.2.1 setup()

```
void Iir::RBJ::HighPass::setup (
    double sampleRate,
    double cutoffFrequency,
    double q )
```

Calculates the coefficients

Parameters

<i>sampleRate</i>	Sampling rate
<i>cutoffFrequency</i>	Cutoff frequency
<i>q</i>	Q factor determines the resonance peak at the cutoff.

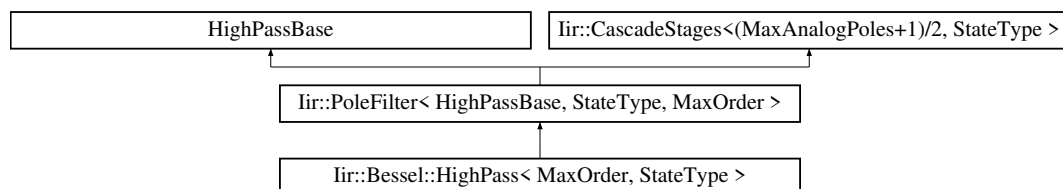
The documentation for this struct was generated from the following files:

- iir/RBJ.h
- iir/RBJ.cpp

6.51 Iir::Bessel::HighPass< MaxOrder, StateType > Struct Template Reference

```
#include <Bessel.h>
```

Inheritance diagram for Iir::Bessel::HighPass< MaxOrder, StateType >:



Public Member Functions

- void [setup](#) (double sampleRate, double cutoffFrequency)

6.51.1 Detailed Description

```
template<int MaxOrder, class StateType = DEFAULT_STATE>
struct Iir::Bessel::HighPass< MaxOrder, StateType >
```

Bessel Highpass.

6.51.2 Member Function Documentation

6.51.2.1 setup()

```
template<int MaxOrder, class StateType = DEFAULT_STATE>
void Iir::Bessel::HighPass< MaxOrder, StateType >::setup (
    double sampleRate,
    double cutoffFrequency ) [inline]
```

Calculate the coefficients

Parameters

<i>sampleRate</i>	Sampling rate
<i>cutoffFrequency</i>	Cutoff frequency

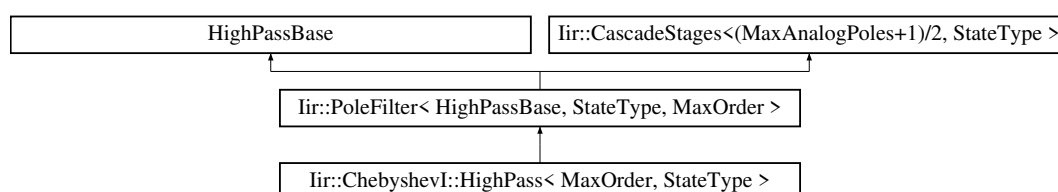
The documentation for this struct was generated from the following file:

- iir/Bessel.h

6.52 Iir::ChebyshevI::HighPass< MaxOrder, StateType > Struct Template Reference

```
#include <ChebyshevI.h>
```

Inheritance diagram for Iir::ChebyshevI::HighPass< MaxOrder, StateType >:



Public Member Functions

- void `setup` (double sampleRate, double cutoffFrequency, double rippleDb)

6.52.1 Detailed Description

```
template<int MaxOrder, class StateType = DEFAULT_STATE>
struct Iir::ChebyshevI::HighPass< MaxOrder, StateType >
```

[ChebyshevI](#) highpass filter

6.52.2 Member Function Documentation

6.52.2.1 setup()

```
template<int MaxOrder, class StateType = DEFAULT_STATE>
void Iir::ChebyshevI::HighPass< MaxOrder, StateType >::setup (
    double sampleRate,
    double cutoffFrequency,
    double rippleDb ) [inline]
```

Calculates the coefficients of the filter

Parameters

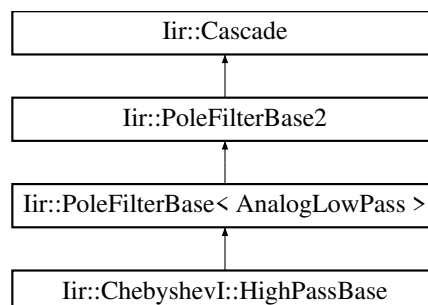
<i>sampleRate</i>	Sampling rate
<i>cutoffFrequency</i>	Cutoff frequency.
<i>rippleDb</i>	Permitted ripples in dB in the passband

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

- iir/ChebyshevI.h

6.53 Iir::ChebyshevI::HighPassBase Struct Reference

Inheritance diagram for Iir::ChebyshevI::HighPassBase:

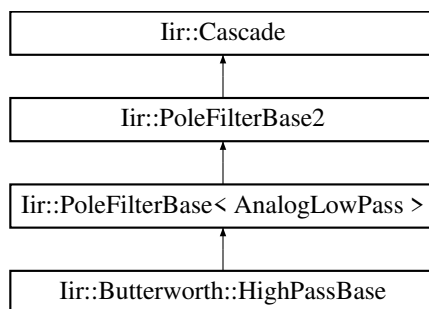


The documentation for this struct was generated from the following files:

- iir/ChebyshevI.h
- iir/ChebyshevI.cpp

6.54 Iir::Butterworth::HighPassBase Struct Reference

Inheritance diagram for Iir::Butterworth::HighPassBase:

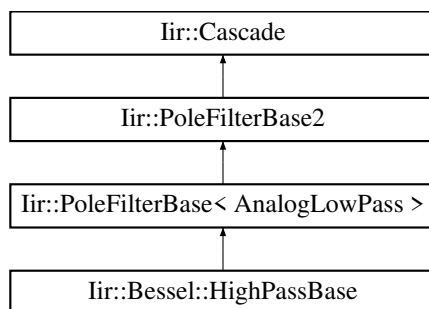


The documentation for this struct was generated from the following files:

- iir/Butterworth.h
- iir/Butterworth.cpp

6.55 Iir::Bessel::HighPassBase Struct Reference

Inheritance diagram for Iir::Bessel::HighPassBase:

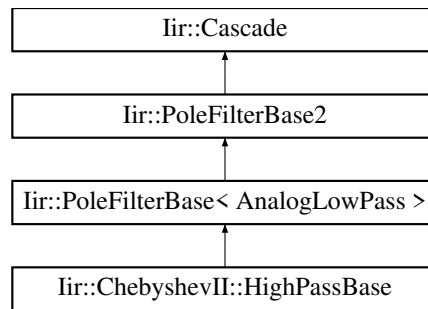


The documentation for this struct was generated from the following files:

- iir/Bessel.h
- iir/Bessel.cpp

6.56 Iir::ChebyshevII::HighPassBase Struct Reference

Inheritance diagram for Iir::ChebyshevII::HighPassBase:



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

- iir/ChebyshevII.h
- iir/ChebyshevII.cpp

6.57 Iir::HighPassTransform Class Reference

```
#include <PoleFilter.h>
```

6.57.1 Detailed Description

low pass to high pass

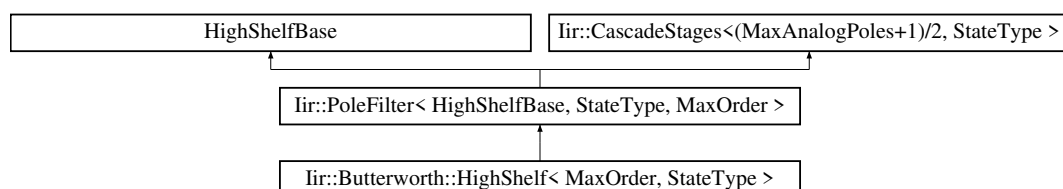
The documentation for this class was generated from the following files:

- iir/PoleFilter.h
- iir/PoleFilter.cpp

6.58 Iir::Butterworth::HighShelf< MaxOrder, StateType > Struct Template Reference

```
#include <Butterworth.h>
```

Inheritance diagram for Iir::Butterworth::HighShelf< MaxOrder, StateType >:



Public Member Functions

- void [setup](#) (double sampleRate, double cutoffFrequency, double gainDb)

6.58.1 Detailed Description

```
template<int MaxOrder, class StateType = DEFAULT_STATE>
struct Iir::Butterworth::HighShelf< MaxOrder, StateType >
```

[Butterworth](#) high shelf filter. Above the cutoff the filter has a specified gain and below it has 0 dB.

6.58.2 Member Function Documentation

6.58.2.1 setup()

```
template<int MaxOrder, class StateType = DEFAULT_STATE>
void Iir::Butterworth::HighShelf< MaxOrder, StateType >::setup (
    double sampleRate,
    double cutoffFrequency,
    double gainDb ) [inline]
```

Calculates the coefficients

Parameters

<i>sampleRate</i>	Sampling rate
<i>cutoffFrequency</i>	Cutoff
<i>gainDb</i>	Gain in dB of the filter in the passband

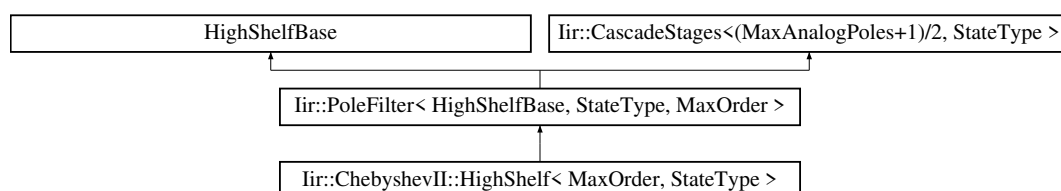
The documentation for this struct was generated from the following file:

- iir/Butterworth.h

6.59 Iir::ChebyshevII::HighShelf< MaxOrder, StateType > Struct Template Reference

```
#include <ChebyshevII.h>
```

Inheritance diagram for Iir::ChebyshevII::HighShelf< MaxOrder, StateType >:



Public Member Functions

- void [setup](#) (double sampleRate, double cutoffFrequency, double gainDb, double stopBandDb)

6.59.1 Detailed Description

```
template<int MaxOrder, class StateType = DEFAULT_STATE>
struct Iir::ChebyshevII::HighShelf< MaxOrder, StateType >
```

[ChebyshevII](#) high shelf filter. Specified gain in the passband and 0dB in the stopband.

6.59.2 Member Function Documentation

6.59.2.1 setup()

```
template<int MaxOrder, class StateType = DEFAULT_STATE>
void Iir::ChebyshevII::HighShelf< MaxOrder, StateType >::setup (
    double sampleRate,
    double cutoffFrequency,
    double gainDb,
    double stopBandDb ) [inline]
```

Calculates the coefficients of the filter

Parameters

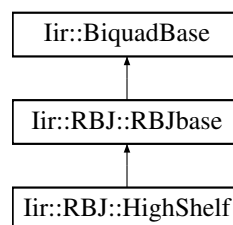
<i>sampleRate</i>	Sampling rate
<i>cutoffFrequency</i>	Cutoff frequency.
<i>gainDb</i>	Gain the passband. The stopband has 0 dB gain.
<i>stopBandDb</i>	Permitted ripples in dB in the stopband

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

- iir/ChebyshevII.h

6.60 Iir::RBJ::HighShelf Struct Reference

Inheritance diagram for Iir::RBJ::HighShelf:



Additional Inherited Members

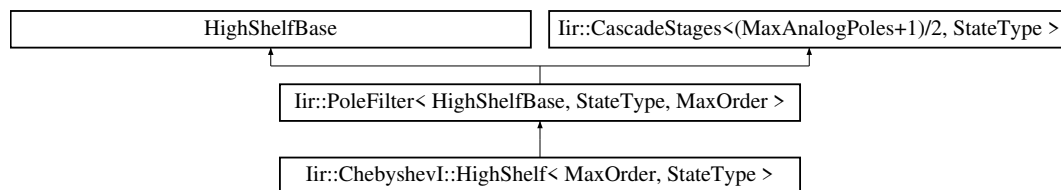
The documentation for this struct was generated from the following files:

- iir/RBJ.h
- iir/RBJ.cpp

6.61 Iir::ChebyshevI::HighShelf< MaxOrder, StateType > Struct Template Reference

```
#include <ChebyshevI.h>
```

Inheritance diagram for Iir::ChebyshevI::HighShelf< MaxOrder, StateType >:



Public Member Functions

- void [setup](#) (double sampleRate, double cutoffFrequency, double gainDb, double rippleDb)

6.61.1 Detailed Description

```
template<int MaxOrder, class StateType = DEFAULT_STATE>
struct Iir::ChebyshevI::HighShelf< MaxOrder, StateType >
```

[ChebyshevI](#) high shelf filter. Specified gain in the passband. Otherwise 0 dB.

6.61.2 Member Function Documentation

6.61.2.1 setup()

```
template<int MaxOrder, class StateType = DEFAULT_STATE>
void Iir::ChebyshevI::HighShelf< MaxOrder, StateType >::setup (
    double sampleRate,
    double cutoffFrequency,
    double gainDb,
    double rippleDb ) [inline]
```

Calculates the coefficients of the filter

Parameters

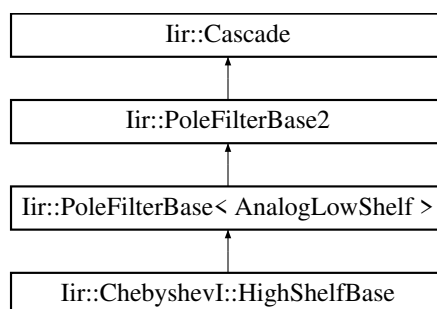
<i>sampleRate</i>	Sampling rate
<i>cutoffFrequency</i>	Cutoff frequency.
<i>gainDb</i>	Gain in the passband
<i>rippleDb</i>	Permitted ripples in dB in the passband

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

- iir/ChebyshevI.h

6.62 Iir::ChebyshevI::HighShelfBase Struct Reference

Inheritance diagram for Iir::ChebyshevI::HighShelfBase:

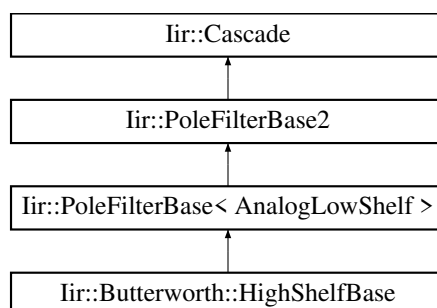


The documentation for this struct was generated from the following files:

- iir/ChebyshevI.h
- iir/ChebyshevI.cpp

6.63 Iir::Butterworth::HighShelfBase Struct Reference

Inheritance diagram for Iir::Butterworth::HighShelfBase:

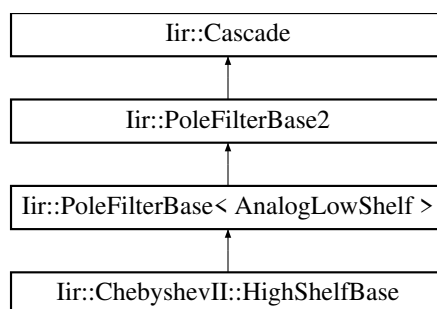


The documentation for this struct was generated from the following files:

- iir/Butterworth.h
- iir/Butterworth.cpp

6.64 Iir::ChebyshevII::HighShelfBase Struct Reference

Inheritance diagram for Iir::ChebyshevII::HighShelfBase:



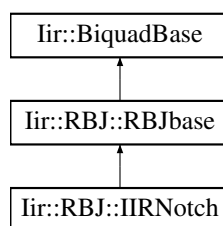
The documentation for this struct was generated from the following files:

- iir/ChebyshevII.h
- iir/ChebyshevII.cpp

6.65 Iir::RBJ::IIRNotch Struct Reference

```
#include <RBJ.h>
```

Inheritance diagram for Iir::RBJ::IIRNotch:



Public Member Functions

- void [setup](#) (double sampleRate, double centerFrequency, double q_factor)

6.65.1 Detailed Description

Bandstop with Q factor: the higher the Q factor the more narrow is the notch. However, a narrow notch has a long impulse response (= ringing) and numerical problems might prevent perfect damping. Practical values of the Q factor are about Q = 10 to 20. In terms of the design the Q factor defines the radius of the poles as $r = \exp(-\pi * (\text{centerFrequency} / \text{sampleRate}) / q_factor)$ whereas the angles of the poles/zeros define the bandstop frequency. The higher Q the closer r moves towards the unit circle.

6.65.2 Member Function Documentation

6.65.2.1 setup()

```
void Iir::RBJ::IIRNotch::setup (
    double sampleRate,
    double centerFrequency,
    double q_factor )
```

Calculates the coefficients

Parameters

<i>sampleRate</i>	Sampling rate
<i>centerFrequency</i>	Center frequency of the notch
<i>q_factor</i>	Q factor of the notch (1 to ~20)

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

- iir/RBJ.h
- iir/RBJ.cpp

6.66 Iir::Layout< MaxPoles > Class Template Reference

```
#include <Layout.h>
```

6.66.1 Detailed Description

```
template<int MaxPoles>
class Iir::Layout< MaxPoles >
```

Storage for [Layout](#)

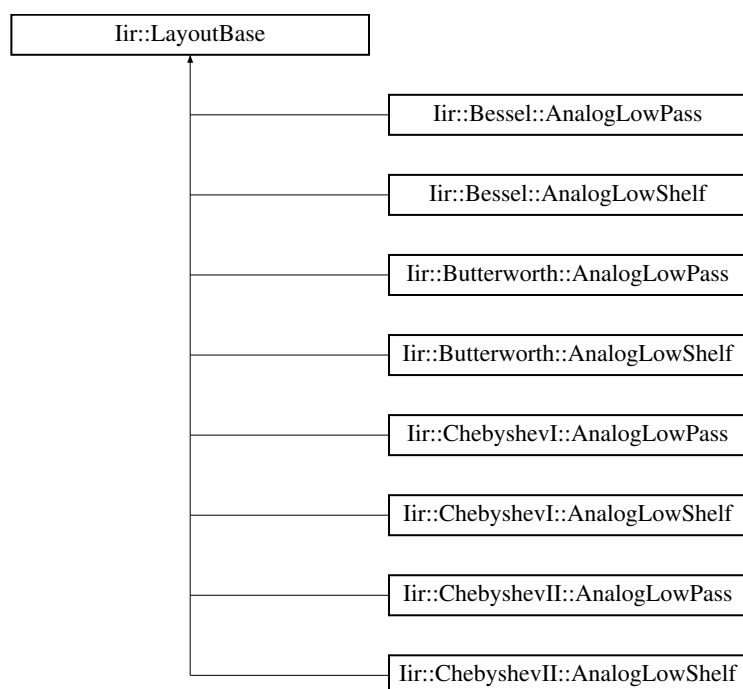
The documentation for this class was generated from the following file:

- iir/Layout.h

6.67 Iir::LayoutBase Class Reference

```
#include <Layout.h>
```

Inheritance diagram for Iir::LayoutBase:



6.67.1 Detailed Description

Base uses pointers to reduce template instantiations

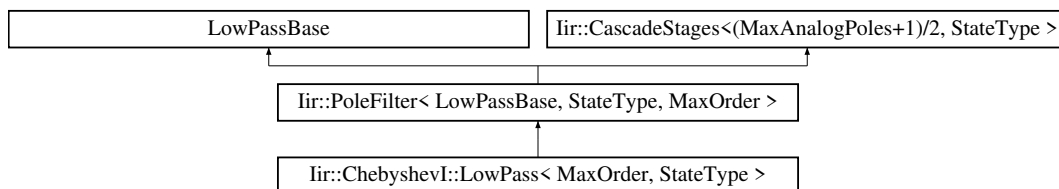
The documentation for this class was generated from the following file:

- iir/Layout.h

6.68 Iir::ChebyshevI::LowPass< MaxOrder, StateType > Struct Template Reference

```
#include <ChebyshevI.h>
```

Inheritance diagram for Iir::ChebyshevI::LowPass< MaxOrder, StateType >:



Public Member Functions

- void [setup](#) (double sampleRate, double cutoffFrequency, double rippleDb)

6.68.1 Detailed Description

```
template<int MaxOrder, class StateType = DEFAULT_STATE>
struct Iir::ChebyshevI::LowPass< MaxOrder, StateType >
```

[ChebyshevI](#) lowpass filter

6.68.2 Member Function Documentation

6.68.2.1 setup()

```
template<int MaxOrder, class StateType = DEFAULT_STATE>
void Iir::ChebyshevI::LowPass< MaxOrder, StateType >::setup (
    double sampleRate,
    double cutoffFrequency,
    double rippleDb ) [inline]
```

Calculates the coefficients of the filter

Parameters

<i>sampleRate</i>	Sampling rate
<i>cutoffFrequency</i>	Cutoff frequency.
<i>rippleDb</i>	Permitted ripples in dB in the passband

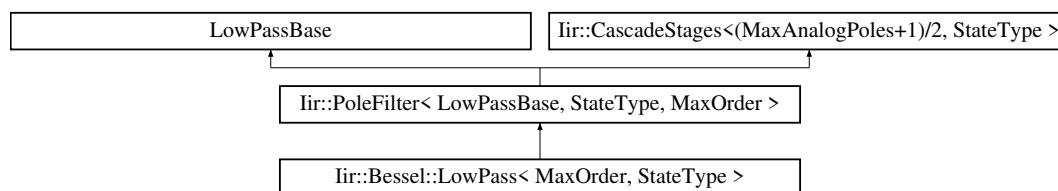
The documentation for this struct was generated from the following file:

- `iir/ChebyshevI.h`

6.69 `lir::Bessel::LowPass< MaxOrder, StateType >` Struct Template Reference

```
#include <Bessel.h>
```

Inheritance diagram for `lir::Bessel::LowPass< MaxOrder, StateType >`:



Public Member Functions

- `void setup (double sampleRate, double cutoffFrequency)`

6.69.1 Detailed Description

```
template<int MaxOrder, class StateType = DEFAULT_STATE>
struct lir::Bessel::LowPass< MaxOrder, StateType >
```

[Bessel](#) Lowpass

6.69.2 Member Function Documentation

6.69.2.1 `setup()`

```
template<int MaxOrder, class StateType = DEFAULT_STATE>
void lir::Bessel::LowPass< MaxOrder, StateType >::setup (
    double sampleRate,
    double cutoffFrequency ) [inline]
```

Calculate the coefficients

Parameters

<i>sampleRate</i>	Sampling rate
<i>cutoffFrequency</i>	Cutoff frequency

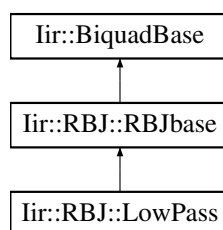
The documentation for this struct was generated from the following file:

- iir/Bessel.h

6.70 Iir::RBJ::LowPass Struct Reference

```
#include <RBJ.h>
```

Inheritance diagram for Iir::RBJ::LowPass:

**Public Member Functions**

- void [setup](#) (double sampleRate, double cutoffFrequency, double q)

6.70.1 Detailed Description

Lowpass.

6.70.2 Member Function Documentation**6.70.2.1 setup()**

```
void Iir::RBJ::LowPass::setup (
    double sampleRate,
    double cutoffFrequency,
    double q )
```

Calculates the coefficients

Parameters

<i>sampleRate</i>	Sampling rate
<i>cutoffFrequency</i>	Cutoff frequency
<i>q</i>	Q factor determines the resonance peak at the cutoff.

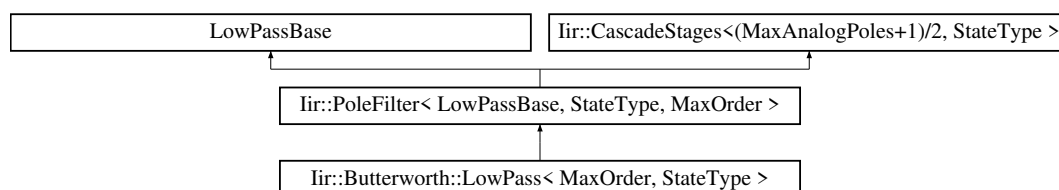
The documentation for this struct was generated from the following files:

- iir/RBJ.h
- iir/RBJ.cpp

6.71 Iir::Butterworth::LowPass< MaxOrder, StateType > Struct Template Reference

```
#include <Butterworth.h>
```

Inheritance diagram for Iir::Butterworth::LowPass< MaxOrder, StateType >:



Public Member Functions

- void [setup](#) (double *sampleRate*, double *cutoffFrequency*)

6.71.1 Detailed Description

```
template<int MaxOrder, class StateType = DEFAULT_STATE>
struct Iir::Butterworth::LowPass< MaxOrder, StateType >
```

[Butterworth](#) Lowpass filter.

6.71.2 Member Function Documentation

6.71.2.1 setup()

```
template<int MaxOrder, class StateType = DEFAULT_STATE>
void Iir::Butterworth::LowPass< MaxOrder, StateType >::setup (
    double sampleRate,
    double cutoffFrequency ) [inline]
```

Calculates the coefficients

Parameters

<i>sampleRate</i>	Sampling rate
<i>cutoffFrequency</i>	Cutoff

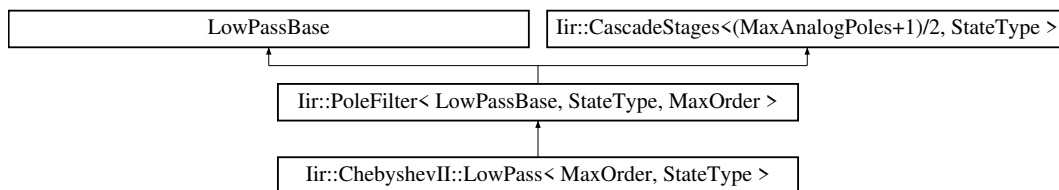
The documentation for this struct was generated from the following file:

- `iir/Butterworth.h`

6.72 `Iir::ChebyshevII::LowPass< MaxOrder, StateType >` Struct Template Reference

```
#include <ChebyshevII.h>
```

Inheritance diagram for `Iir::ChebyshevII::LowPass< MaxOrder, StateType >`:



Public Member Functions

- `void setup` (double `sampleRate`, double `cutoffFrequency`, double `stopBandDb`)

6.72.1 Detailed Description

```
template<int MaxOrder, class StateType = DEFAULT_STATE>
struct Iir::ChebyshevII::LowPass< MaxOrder, StateType >
```

[ChebyshevII](#) lowpass filter

6.72.2 Member Function Documentation

6.72.2.1 `setup()`

```
template<int MaxOrder, class StateType = DEFAULT_STATE>
void Iir::ChebyshevII::LowPass< MaxOrder, StateType >::setup (
    double sampleRate,
    double cutoffFrequency,
    double stopBandDb ) [inline]
```

Calculates the coefficients of the filter

Parameters

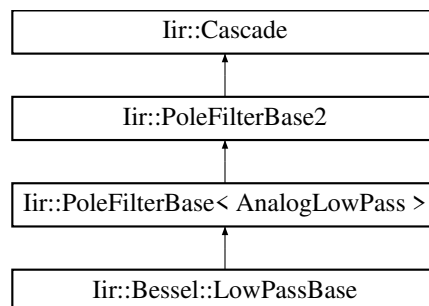
<i>sampleRate</i>	Sampling rate
<i>cutoffFrequency</i>	Cutoff frequency.
<i>stopBandDb</i>	Permitted ripples in dB in the stopband

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

- iir/ChebyshevII.h

6.73 Iir::Bessel::LowPassBase Struct Reference

Inheritance diagram for Iir::Bessel::LowPassBase:

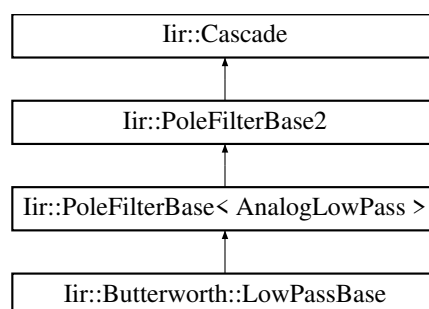


The documentation for this struct was generated from the following files:

- iir/Bessel.h
- iir/Bessel.cpp

6.74 Iir::Butterworth::LowPassBase Struct Reference

Inheritance diagram for Iir::Butterworth::LowPassBase:

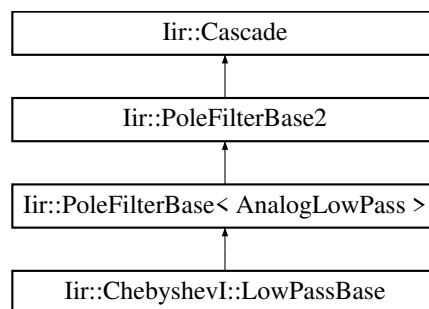


The documentation for this struct was generated from the following files:

- iir/Butterworth.h
- iir/Butterworth.cpp

6.75 Iir::ChebyshevI::LowPassBase Struct Reference

Inheritance diagram for Iir::ChebyshevI::LowPassBase:

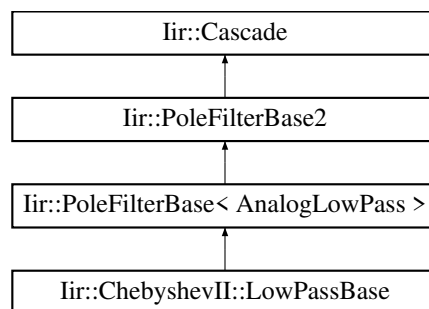


The documentation for this struct was generated from the following files:

- iir/ChebyshevI.h
- iir/ChebyshevI.cpp

6.76 Iir::ChebyshevII::LowPassBase Struct Reference

Inheritance diagram for Iir::ChebyshevII::LowPassBase:



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

- iir/ChebyshevII.h
- iir/ChebyshevII.cpp

6.77 Iir::LowPassTransform Class Reference

```
#include <PoleFilter.h>
```

6.77.1 Detailed Description

s-plane to z-plane transforms

For pole filters, an analog prototype is created via placement of poles and zeros in the s-plane. The analog prototype is either a halfband low pass or a halfband low shelf. The poles, zeros, and normalization parameters are transformed into the z-plane using variants of the bilinear transformation. low pass to low pass

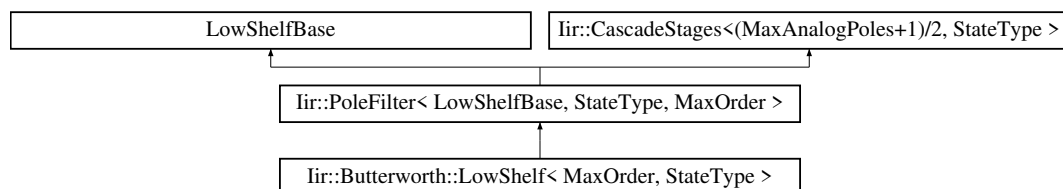
The documentation for this class was generated from the following files:

- iir/PoleFilter.h
- iir/PoleFilter.cpp

6.78 Iir::Butterworth::LowShelf< MaxOrder, StateType > Struct Template Reference

```
#include <Butterworth.h>
```

Inheritance diagram for Iir::Butterworth::LowShelf< MaxOrder, StateType >:



Public Member Functions

- void `setup` (double `sampleRate`, double `cutoffFrequency`, double `gainDb`)

6.78.1 Detailed Description

```
template<int MaxOrder, class StateType = DEFAULT_STATE>
struct Iir::Butterworth::LowShelf< MaxOrder, StateType >
```

[Butterworth](#) low shelf filter: below the cutoff it has a specified gain and above the cutoff the gain is 0 dB.

6.78.2 Member Function Documentation

6.78.2.1 `setup()`

```
template<int MaxOrder, class StateType = DEFAULT_STATE>
void Iir::Butterworth::LowShelf< MaxOrder, StateType >::setup (
    double sampleRate,
    double cutoffFrequency,
    double gainDb ) [inline]
```

Calculates the coefficients

Parameters

<i>sampleRate</i>	Sampling rate
<i>cutoffFrequency</i>	Cutoff
<i>gainDb</i>	Gain in dB of the filter in the passband

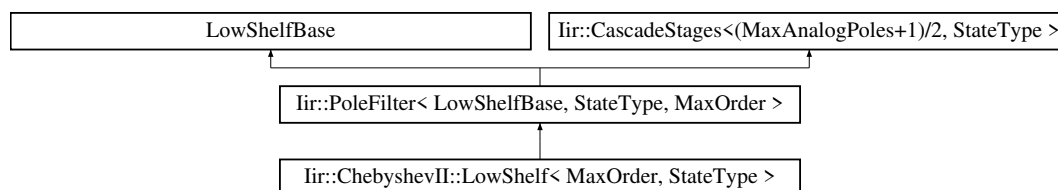
The documentation for this struct was generated from the following file:

- iir/Butterworth.h

6.79 Iir::ChebyshevII::LowShelf< MaxOrder, StateType > Struct Template Reference

```
#include <ChebyshevII.h>
```

Inheritance diagram for Iir::ChebyshevII::LowShelf< MaxOrder, StateType >:



Public Member Functions

- void [setup](#) (double sampleRate, double cutoffFrequency, double gainDb, double stopBandDb)

6.79.1 Detailed Description

```
template<int MaxOrder, class StateType = DEFAULT_STATE>
struct Iir::ChebyshevII::LowShelf< MaxOrder, StateType >
```

[ChebyshevII](#) low shelf filter. Specified gain in the passband and 0dB in the stopband.

6.79.2 Member Function Documentation

6.79.2.1 setup()

```
template<int MaxOrder, class StateType = DEFAULT_STATE>
void Iir::ChebyshevII::LowShelf< MaxOrder, StateType >::setup (
    double sampleRate,
    double cutoffFrequency,
    double gainDb,
    double stopBandDb ) [inline]
```

Calculates the coefficients of the filter

Parameters

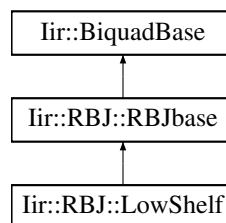
<i>sampleRate</i>	Sampling rate
<i>cutoffFrequency</i>	Cutoff frequency.
<i>gainDb</i>	Gain the passband. The stopband has 0 dB gain.
<i>stopBandDb</i>	Permitted ripples in dB in the stopband

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

- `iir/ChebyshevII.h`

6.80 `lir::RBJ::LowShelf` Struct Reference

Inheritance diagram for `lir::RBJ::LowShelf`:



Additional Inherited Members

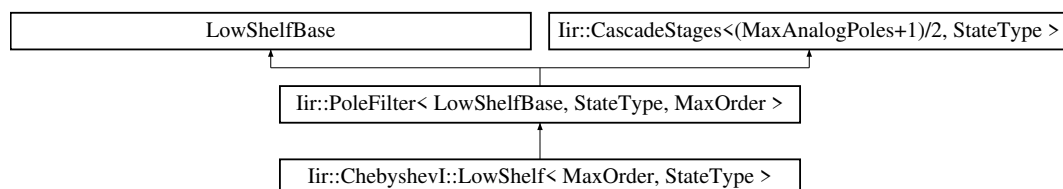
The documentation for this struct was generated from the following files:

- `iir/RBJ.h`
- `iir/RBJ.cpp`

6.81 `lir::ChebyshevI::LowShelf< MaxOrder, StateType >` Struct Template Reference

```
#include <ChebyshevI.h>
```

Inheritance diagram for `lir::ChebyshevI::LowShelf< MaxOrder, StateType >`:



Public Member Functions

- `void setup` (double sampleRate, double cutoffFrequency, double gainDb, double rippleDb)

6.81.1 Detailed Description

```
template<int MaxOrder, class StateType = DEFAULT_STATE>
struct Iir::ChebyshevI::LowShelf< MaxOrder, StateType >
```

[ChebyshevI](#) low shelf filter. Specified gain in the passband. Otherwise 0 dB.

6.81.2 Member Function Documentation

6.81.2.1 setup()

```
template<int MaxOrder, class StateType = DEFAULT_STATE>
void Iir::ChebyshevI::LowShelf< MaxOrder, StateType >::setup (
    double sampleRate,
    double cutoffFrequency,
    double gainDb,
    double rippleDb ) [inline]
```

Calculates the coefficients of the filter

Parameters

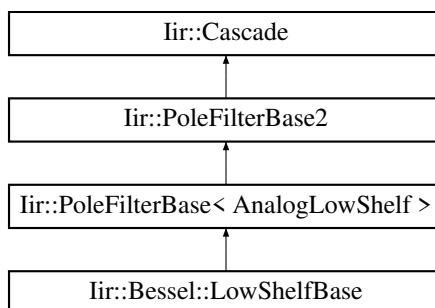
<i>sampleRate</i>	Sampling rate
<i>cutoffFrequency</i>	Cutoff frequency.
<i>gainDb</i>	Gain in the passband
<i>rippleDb</i>	Permitted ripples in dB in the passband

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

- [iir/ChebyshevI.h](#)

6.82 Iir::Bessel::LowShelfBase Struct Reference

Inheritance diagram for Iir::Bessel::LowShelfBase:

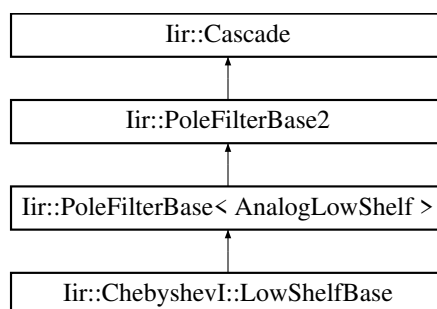


The documentation for this struct was generated from the following files:

- [iir/Bessel.h](#)
- [iir/Bessel.cpp](#)

6.83 Iir::ChebyshevI::LowShelfBase Struct Reference

Inheritance diagram for Iir::ChebyshevI::LowShelfBase:

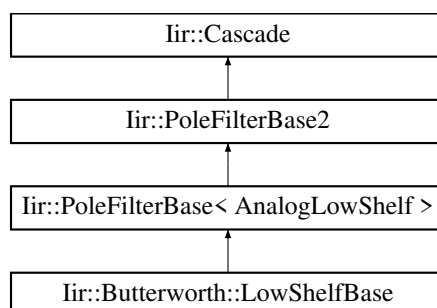


The documentation for this struct was generated from the following files:

- iir/ChebyshevI.h
- iir/ChebyshevI.cpp

6.84 Iir::Butterworth::LowShelfBase Struct Reference

Inheritance diagram for Iir::Butterworth::LowShelfBase:

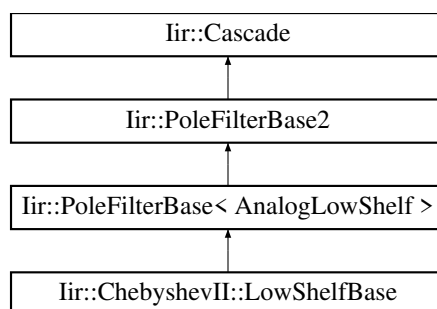


The documentation for this struct was generated from the following files:

- iir/Butterworth.h
- iir/Butterworth.cpp

6.85 Iir::ChebyshevII::LowShelfBase Struct Reference

Inheritance diagram for Iir::ChebyshevII::LowShelfBase:

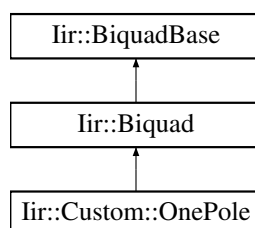


The documentation for this struct was generated from the following files:

- iir/ChebyshevII.h
- iir/ChebyshevII.cpp

6.86 Iir::Custom::OnePole Struct Reference

Inheritance diagram for Iir::Custom::OnePole:



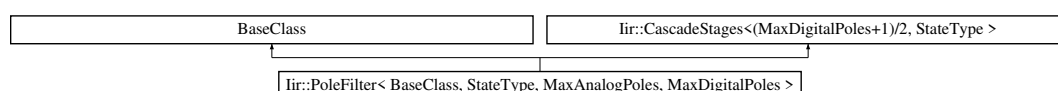
The documentation for this struct was generated from the following files:

- iir/Custom.h
- iir/Custom.cpp

6.87 Iir::PoleFilter< BaseClass, StateType, MaxAnalogPoles, MaxDigitalPoles > Struct Template Reference

```
#include <PoleFilter.h>
```

Inheritance diagram for Iir::PoleFilter< BaseClass, StateType, MaxAnalogPoles, MaxDigitalPoles >:



6.87.1 Detailed Description

```
template<class BaseClass, class StateType, int MaxAnalogPoles, int MaxDigitalPoles = MaxAnalogPoles>
struct Iir::PoleFilter< BaseClass, StateType, MaxAnalogPoles, MaxDigitalPoles >
```

Storage for pole filters

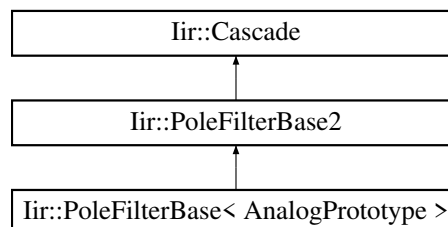
The documentation for this struct was generated from the following file:

- iir/PoleFilter.h

6.88 Iir::PoleFilterBase< AnalogPrototype > Class Template Reference

```
#include <PoleFilter.h>
```

Inheritance diagram for Iir::PoleFilterBase< AnalogPrototype >:



6.88.1 Detailed Description

```
template<class AnalogPrototype>
class Iir::PoleFilterBase< AnalogPrototype >
```

Serves a container to hold the analog prototype and the digital pole/zero layout.

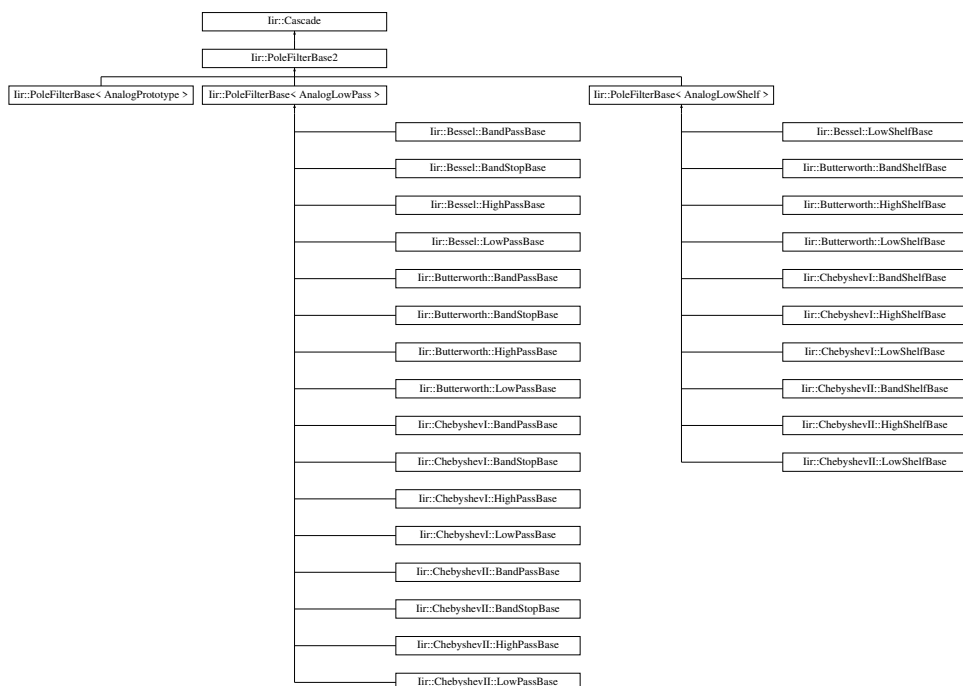
The documentation for this class was generated from the following file:

- iir/PoleFilter.h

6.89 Iir::PoleFilterBase2 Class Reference

```
#include <PoleFilter.h>
```

Inheritance diagram for Iir::PoleFilterBase2:



6.89.1 Detailed Description

Factored implementations to reduce template instantiations

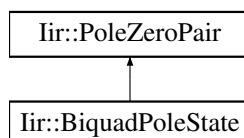
The documentation for this class was generated from the following file:

- iir/PoleFilter.h

6.90 Iir::PoleZeroPair Struct Reference

```
#include <Types.h>
```

Inheritance diagram for Iir::PoleZeroPair:



6.90.1 Detailed Description

A pair of pole/zeros. This fits in a biquad (but is missing the gain)

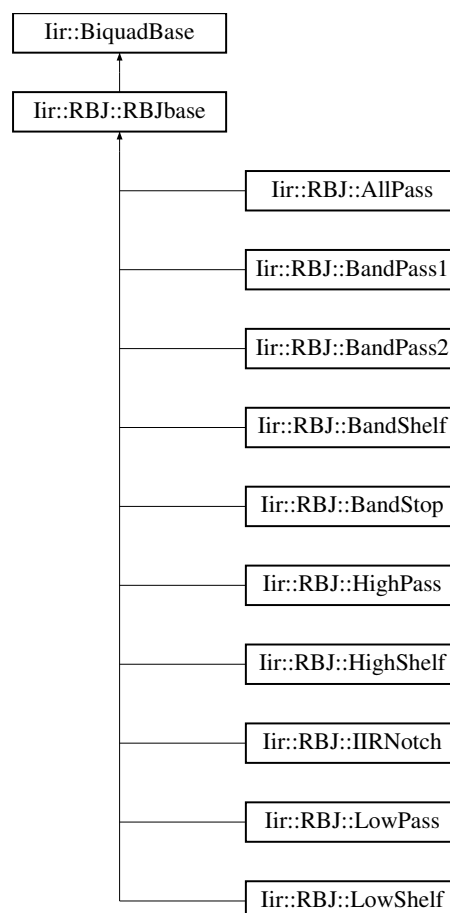
The documentation for this struct was generated from the following file:

- iir/Types.h

6.91 Iir::RBJ::RBJbase Struct Reference

```
#include <RBJ.h>
```

Inheritance diagram for Iir::RBJ::RBJbase:



Public Member Functions

- template<typename Sample >
Sample [filter](#) (Sample s)
filter operation
- void [reset](#) ()
resets the delay lines to zero
- const [DirectFormI](#) & [getState](#) ()
gets the delay lines (=state) of the filter

6.91.1 Detailed Description

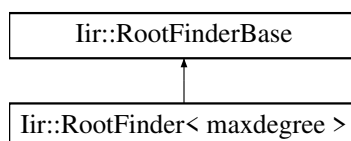
The base class of all [RBJ](#) filters

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

- [iir/RBJ.h](#)

6.92 Iir::RootFinder< maxdegree > Struct Template Reference

Inheritance diagram for Iir::RootFinder< maxdegree >:



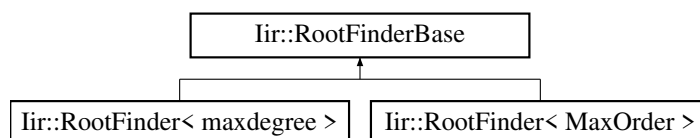
The documentation for this struct was generated from the following file:

- [iir/RootFinder.h](#)

6.93 Iir::RootFinderBase Class Reference

```
#include <RootFinder.h>
```

Inheritance diagram for Iir::RootFinderBase:



Classes

- struct [Array](#)

6.93.1 Detailed Description

Finds the complex roots of the given polynomial with complex-valued coefficients using a numerical method.

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

- [iir/RootFinder.h](#)
- [iir/RootFinder.cpp](#)

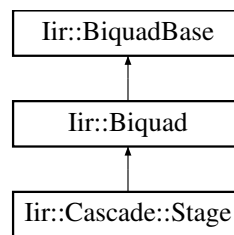
6.94 Iir::SlopeDetector< Channels, Value > Class Template Reference

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

- iir/Utilities.h

6.95 Iir::Cascade::Stage Struct Reference

Inheritance diagram for Iir::Cascade::Stage:

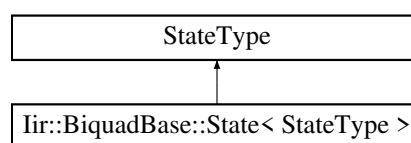


The documentation for this struct was generated from the following file:

- iir/Cascade.h

6.96 Iir::BiquadBase::State< StateType > Struct Template Reference

Inheritance diagram for Iir::BiquadBase::State< StateType >:



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

- iir/Biquad.h

6.97 Iir::Cascade::Storage Struct Reference

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

- iir/Cascade.h

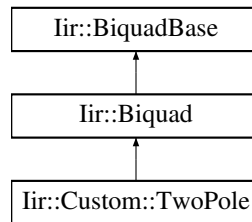
6.98 Iir::TransposedDirectFormII Class Reference

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

- iir/State.h

6.99 Iir::Custom::TwoPole Struct Reference

Inheritance diagram for Iir::Custom::TwoPole:

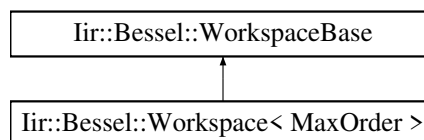


The documentation for this struct was generated from the following files:

- iir/Custom.h
- iir/Custom.cpp

6.100 Iir::Bessel::Workspace< MaxOrder > Struct Template Reference

Inheritance diagram for Iir::Bessel::Workspace< MaxOrder >:

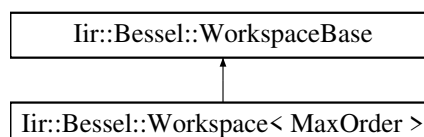


The documentation for this struct was generated from the following file:

- iir/Bessel.h

6.101 Iir::Bessel::WorkspaceBase Struct Reference

Inheritance diagram for Iir::Bessel::WorkspaceBase:



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

- iir/Bessel.h

Index

- add
 - [lir, 14, 15](#)
- copy
 - [lir, 15](#)
- [lir, 12](#)
 - [add, 14, 15](#)
 - [copy, 15](#)
 - [Kind, 14](#)
- [lir::BandPassTransform, 29](#)
- [lir::BandStopTransform, 40](#)
- [lir::Bessel, 15](#)
- [lir::Bessel::AnalogLowPass, 18](#)
- [lir::Bessel::AnalogLowShelf, 20](#)
- [lir::Bessel::BandPass](#)
 - [setup, 23](#)
- [lir::Bessel::BandPass< MaxOrder, StateType >, 22](#)
- [lir::Bessel::BandPassBase, 28](#)
- [lir::Bessel::BandStop](#)
 - [setup, 37](#)
- [lir::Bessel::BandStop< MaxOrder, StateType >, 37](#)
- [lir::Bessel::BandStopBase, 39](#)
- [lir::Bessel::HighPass](#)
 - [setup, 48](#)
- [lir::Bessel::HighPass< MaxOrder, StateType >, 47](#)
- [lir::Bessel::HighPassBase, 50](#)
- [lir::Bessel::LowPass](#)
 - [setup, 59](#)
- [lir::Bessel::LowPass< MaxOrder, StateType >, 59](#)
- [lir::Bessel::LowPassBase, 63](#)
- [lir::Bessel::LowShelfBase, 68](#)
- [lir::Bessel::Workspace< MaxOrder >, 76](#)
- [lir::Bessel::WorkspaceBase, 76](#)
- [lir::Biquad, 41](#)
- [lir::BiquadBase, 41](#)
- [lir::BiquadBase::State< StateType >, 75](#)
- [lir::BiquadPoleState, 42](#)
- [lir::Butterworth, 16](#)
- [lir::Butterworth::AnalogLowPass, 19](#)
- [lir::Butterworth::AnalogLowShelf, 20](#)
- [lir::Butterworth::BandPass](#)
 - [setup, 25](#)
- [lir::Butterworth::BandPass< MaxOrder, StateType >, 24](#)
- [lir::Butterworth::BandPassBase, 27](#)
- [lir::Butterworth::BandShelf](#)
 - [setup, 30](#)
- [lir::Butterworth::BandShelf< MaxOrder, StateType >, 29](#)
- [lir::Butterworth::BandShelfBase, 33](#)
- [lir::Butterworth::BandStop](#)
 - [setup, 38](#)
- [lir::Butterworth::BandStop< MaxOrder, StateType >, 38](#)
- [lir::Butterworth::BandStopBase, 40](#)
- [lir::Butterworth::HighPass](#)
 - [setup, 46](#)
- [lir::Butterworth::HighPass< MaxOrder, StateType >, 45](#)
- [lir::Butterworth::HighPassBase, 50](#)
- [lir::Butterworth::HighShelf](#)
 - [setup, 52](#)
- [lir::Butterworth::HighShelf< MaxOrder, StateType >, 51](#)
- [lir::Butterworth::HighShelfBase, 55](#)
- [lir::Butterworth::LowPass](#)
 - [setup, 61](#)
- [lir::Butterworth::LowPass< MaxOrder, StateType >, 61](#)
- [lir::Butterworth::LowPassBase, 63](#)
- [lir::Butterworth::LowShelf](#)
 - [setup, 65](#)
- [lir::Butterworth::LowShelf< MaxOrder, StateType >, 65](#)
- [lir::Butterworth::LowShelfBase, 69](#)
- [lir::Cascade, 42](#)
- [lir::Cascade::Stage, 75](#)
- [lir::Cascade::Storage, 75](#)
- [lir::CascadeStages< MaxStages, StateType >, 43](#)
- [lir::ChebyshevI::AnalogLowPass, 19](#)
- [lir::ChebyshevI::AnalogLowShelf, 20](#)
- [lir::ChebyshevI::BandPass](#)
 - [setup, 24](#)
- [lir::ChebyshevI::BandPass< MaxOrder, StateType >, 23](#)
- [lir::ChebyshevI::BandPassBase, 28](#)
- [lir::ChebyshevI::BandShelf](#)
 - [setup, 32](#)
- [lir::ChebyshevI::BandShelf< MaxOrder, StateType >, 32](#)
- [lir::ChebyshevI::BandShelfBase, 33](#)
- [lir::ChebyshevI::BandStop](#)
 - [setup, 36](#)
- [lir::ChebyshevI::BandStop< MaxOrder, StateType >, 36](#)
- [lir::ChebyshevI::BandStopBase, 39](#)
- [lir::ChebyshevI::HighPass](#)
 - [setup, 49](#)
- [lir::ChebyshevI::HighPass< MaxOrder, StateType >, 48](#)
- [lir::ChebyshevI::HighPassBase, 49](#)
- [lir::ChebyshevI::HighShelf](#)
 - [setup, 54](#)
- [lir::ChebyshevI::HighShelf< MaxOrder, StateType >, 54](#)
- [lir::ChebyshevI::HighShelfBase, 55](#)
- [lir::ChebyshevI::LowPass](#)
 - [setup, 58](#)
- [lir::ChebyshevI::LowPass< MaxOrder, StateType >, 58](#)
- [lir::ChebyshevI::LowPassBase, 64](#)
- [lir::ChebyshevI::LowShelf](#)
 - [setup, 68](#)
- [lir::ChebyshevI::LowShelf< MaxOrder, StateType >, 67](#)
- [lir::ChebyshevI::LowShelfBase, 69](#)
- [lir::ChebyshevII::AnalogLowPass, 19](#)
- [lir::ChebyshevII::AnalogLowShelf, 21](#)
- [lir::ChebyshevII::BandPass](#)
 - [setup, 22](#)

- lir::ChebyshevII::BandPass< MaxOrder, StateType >, 21
 - lir::ChebyshevII::BandPassBase, 28
 - lir::ChebyshevII::BandShelf
 - setup, 31
 - lir::ChebyshevII::BandShelf< MaxOrder, StateType >, 30
 - lir::ChebyshevII::BandShelfBase, 34
 - lir::ChebyshevII::BandStop
 - setup, 34
 - lir::ChebyshevII::BandStop< MaxOrder, StateType >, 34
 - lir::ChebyshevII::BandStopBase, 40
 - lir::ChebyshevII::HighPass
 - setup, 45
 - lir::ChebyshevII::HighPass< MaxOrder, StateType >, 44
 - lir::ChebyshevII::HighPassBase, 50
 - lir::ChebyshevII::HighShelf
 - setup, 53
 - lir::ChebyshevII::HighShelf< MaxOrder, StateType >, 52
 - lir::ChebyshevII::HighShelfBase, 55
 - lir::ChebyshevII::LowPass
 - setup, 62
 - lir::ChebyshevII::LowPass< MaxOrder, StateType >, 62
 - lir::ChebyshevII::LowPassBase, 64
 - lir::ChebyshevII::LowShelf
 - setup, 66
 - lir::ChebyshevII::LowShelf< MaxOrder, StateType >, 66
 - lir::ChebyshevII::LowShelfBase, 69
 - lir::ChebyshevII, 17
 - lir::ChebyshevI, 16
 - lir::ComplexPair, 43
 - lir::Custom::OnePole, 70
 - lir::Custom::TwoPole, 76
 - lir::DirectFormII, 44
 - lir::DirectFormI, 43
 - lir::EnvelopeFollower< Channels, Value >, 44
 - lir::HighPassTransform, 51
 - lir::Layout< MaxPoles >, 57
 - lir::LayoutBase, 57
 - lir::LowPassTransform, 64
 - lir::PoleFilter< BaseClass, StateType, MaxAnalogPoles, MaxDigitalPoles >, 70
 - lir::PoleFilterBase< AnalogPrototype >, 71
 - lir::PoleFilterBase2, 72
 - lir::PoleZeroPair, 72
 - lir::RBJ::AllPass, 18
 - lir::RBJ::BandPass1, 25
 - setup, 26
 - lir::RBJ::BandPass2, 26
 - setup, 27
 - lir::RBJ::BandShelf, 31
 - lir::RBJ::BandStop, 35
 - setup, 35
 - lir::RBJ::HighPass, 46
 - setup, 47
 - lir::RBJ::HighShelf, 53
 - lir::RBJ::IIRNotch, 56
 - setup, 56
 - lir::RBJ::LowPass, 60
 - setup, 60
 - lir::RBJ::LowShelf, 67
 - lir::RBJ::RBJbase, 73
 - lir::RBJ, 17
 - lir::RootFinder< maxdegree >, 74
 - lir::RootFinderBase, 74
 - lir::RootFinderBase::Array, 21
 - lir::SlopeDetector< Channels, Value >, 75
 - lir::TransposedDirectFormII, 76
- Kind
- lir, 14
- setup
- lir::Bessel::BandPass, 23
 - lir::Bessel::BandStop, 37
 - lir::Bessel::HighPass, 48
 - lir::Bessel::LowPass, 59
 - lir::Butterworth::BandPass, 25
 - lir::Butterworth::BandShelf, 30
 - lir::Butterworth::BandStop, 38
 - lir::Butterworth::HighPass, 46
 - lir::Butterworth::HighShelf, 52
 - lir::Butterworth::LowPass, 61
 - lir::Butterworth::LowShelf, 65
 - lir::ChebyshevI::BandPass, 24
 - lir::ChebyshevI::BandShelf, 32
 - lir::ChebyshevI::BandStop, 36
 - lir::ChebyshevI::HighPass, 49
 - lir::ChebyshevI::HighShelf, 54
 - lir::ChebyshevI::LowPass, 58
 - lir::ChebyshevI::LowShelf, 68
 - lir::ChebyshevII::BandPass, 22
 - lir::ChebyshevII::BandShelf, 31
 - lir::ChebyshevII::BandStop, 34
 - lir::ChebyshevII::HighPass, 45
 - lir::ChebyshevII::HighShelf, 53
 - lir::ChebyshevII::LowPass, 62
 - lir::ChebyshevII::LowShelf, 66
 - lir::RBJ::BandPass1, 26
 - lir::RBJ::BandPass2, 27
 - lir::RBJ::BandStop, 35
 - lir::RBJ::HighPass, 47
 - lir::RBJ::IIRNotch, 56
 - lir::RBJ::LowPass, 60