

iir1

Generated by Doxygen 1.8.13

Contents

1	IIR1 -- Realtime C++ filter library	2
2	Namespace Index	5
2.1	Namespace List	5
3	Hierarchical Index	5
3.1	Class Hierarchy	5
4	Class Index	9
4.1	Class List	9
5	Namespace Documentation	13
5.1	lir Namespace Reference	13
5.1.1	Detailed Description	14
5.1.2	Enumeration Type Documentation	15
5.1.3	Function Documentation	15
5.2	lir::Bessel Namespace Reference	16
5.2.1	Detailed Description	16
5.3	lir::Butterworth Namespace Reference	16
5.3.1	Detailed Description	17
5.4	lir::ChebyshevI Namespace Reference	17
5.4.1	Detailed Description	17
5.5	lir::ChebyshevII Namespace Reference	17
5.5.1	Detailed Description	18
5.6	lir::RBJ Namespace Reference	18
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	19
6.3	lir::ChebyshevI::AnalogLowPass Class Reference	19
6.4	lir::Butterworth::AnalogLowPass Class Reference	19
6.4.1	Detailed Description	20
6.5	lir::ChebyshevII::AnalogLowPass Class Reference	20
6.6	lir::Bessel::AnalogLowShelf Class Reference	20
6.7	lir::Butterworth::AnalogLowShelf Class Reference	20
6.8	lir::ChebyshevI::AnalogLowShelf Class Reference	21
6.9	lir::ChebyshevII::AnalogLowShelf Class Reference	21
6.10	lir::RootFinderBase::Array Struct Reference	21
6.11	lir::ChebyshevI::BandPass< FilterOrder, StateType > Struct Template Reference	21
6.11.1	Detailed Description	22
6.11.2	Member Function Documentation	22
6.12	lir::Butterworth::BandPass< FilterOrder, StateType > Struct Template Reference	22
6.12.1	Detailed Description	23
6.12.2	Member Function Documentation	23
6.13	lir::Bessel::BandPass< FilterOrder, StateType > Struct Template Reference	24
6.13.1	Detailed Description	24
6.13.2	Member Function Documentation	24
6.14	lir::ChebyshevII::BandPass< FilterOrder, StateType > Struct Template Reference	25
6.14.1	Detailed Description	25
6.14.2	Member Function Documentation	25
6.15	lir::RBJ::BandPass1 Struct Reference	26
6.15.1	Detailed Description	26
6.15.2	Member Function Documentation	26
6.16	lir::RBJ::BandPass2 Struct Reference	27
6.16.1	Detailed Description	27
6.16.2	Member Function Documentation	27

6.17	lir::ChebyshevI::BandPassBase Struct Reference	28
6.18	lir::Butterworth::BandPassBase Struct Reference	28
6.19	lir::Bessel::BandPassBase Struct Reference	29
6.20	lir::ChebyshevII::BandPassBase Struct Reference	29
6.21	lir::BandPassTransform Class Reference	29
6.21.1	Detailed Description	29
6.22	lir::ChebyshevII::BandShelf< FilterOrder, StateType > Struct Template Reference	30
6.22.1	Detailed Description	30
6.22.2	Member Function Documentation	30
6.23	lir::RBJ::BandShelf Struct Reference	31
6.24	lir::ChebyshevI::BandShelf< FilterOrder, StateType > Struct Template Reference	31
6.24.1	Detailed Description	31
6.24.2	Member Function Documentation	32
6.25	lir::Butterworth::BandShelf< FilterOrder, StateType > Struct Template Reference	32
6.25.1	Detailed Description	33
6.25.2	Member Function Documentation	33
6.26	lir::ChebyshevI::BandShelfBase Struct Reference	33
6.27	lir::Butterworth::BandShelfBase Struct Reference	34
6.28	lir::ChebyshevII::BandShelfBase Struct Reference	34
6.29	lir::RBJ::BandStop Struct Reference	35
6.29.1	Detailed Description	35
6.29.2	Member Function Documentation	35
6.30	lir::ChebyshevI::BandStop< FilterOrder, StateType > Struct Template Reference	36
6.30.1	Detailed Description	36
6.30.2	Member Function Documentation	36
6.31	lir::Bessel::BandStop< FilterOrder, StateType > Struct Template Reference	37
6.31.1	Detailed Description	37
6.31.2	Member Function Documentation	37
6.32	lir::Butterworth::BandStop< FilterOrder, StateType > Struct Template Reference	38
6.32.1	Detailed Description	38

6.32.2	Member Function Documentation	38
6.33	lir::ChebyshevII::BandStop< FilterOrder, StateType > Struct Template Reference	39
6.33.1	Detailed Description	39
6.33.2	Member Function Documentation	39
6.34	lir::Bessel::BandStopBase Struct Reference	40
6.35	lir::ChebyshevI::BandStopBase Struct Reference	40
6.36	lir::Butterworth::BandStopBase Struct Reference	41
6.37	lir::ChebyshevII::BandStopBase Struct Reference	41
6.38	lir::BandStopTransform Class Reference	42
6.38.1	Detailed Description	42
6.39	lir::Biquad Class Reference	42
6.39.1	Member Function Documentation	43
6.40	lir::BiquadPoleState Struct Reference	46
6.40.1	Detailed Description	46
6.41	lir::Cascade Class Reference	46
6.42	lir::CascadeStages< MaxStages, StateType > Class Template Reference	47
6.42.1	Detailed Description	47
6.42.2	Member Function Documentation	47
6.43	lir::ComplexPair Struct Reference	49
6.43.1	Detailed Description	49
6.44	lir::DirectFormI Class Reference	50
6.44.1	Detailed Description	50
6.45	lir::DirectFormII Class Reference	50
6.45.1	Detailed Description	50
6.46	lir::EnvelopeFollower< Channels, Value > Class Template Reference	50
6.47	lir::RBJ::HighPass Struct Reference	51
6.47.1	Detailed Description	51
6.47.2	Member Function Documentation	51
6.48	lir::Bessel::HighPass< FilterOrder, StateType > Struct Template Reference	52
6.48.1	Detailed Description	52

6.48.2	Member Function Documentation	52
6.49	lir::ChebyshevI::HighPass< FilterOrder, StateType > Struct Template Reference	53
6.49.1	Detailed Description	53
6.49.2	Member Function Documentation	53
6.50	lir::Butterworth::HighPass< FilterOrder, StateType > Struct Template Reference	54
6.50.1	Detailed Description	54
6.50.2	Member Function Documentation	54
6.51	lir::ChebyshevII::HighPass< FilterOrder, StateType > Struct Template Reference	55
6.51.1	Detailed Description	55
6.51.2	Member Function Documentation	56
6.52	lir::Butterworth::HighPassBase Struct Reference	56
6.53	lir::ChebyshevI::HighPassBase Struct Reference	57
6.54	lir::Bessel::HighPassBase Struct Reference	57
6.55	lir::ChebyshevII::HighPassBase Struct Reference	57
6.56	lir::HighPassTransform Class Reference	58
6.56.1	Detailed Description	58
6.57	lir::ChebyshevII::HighShelf< FilterOrder, StateType > Struct Template Reference	58
6.57.1	Detailed Description	58
6.57.2	Member Function Documentation	59
6.58	lir::Butterworth::HighShelf< FilterOrder, StateType > Struct Template Reference	59
6.58.1	Detailed Description	60
6.58.2	Member Function Documentation	60
6.59	lir::RBJ::HighShelf Struct Reference	60
6.60	lir::ChebyshevI::HighShelf< FilterOrder, StateType > Struct Template Reference	61
6.60.1	Detailed Description	61
6.60.2	Member Function Documentation	61
6.61	lir::ChebyshevII::HighShelfBase Struct Reference	62
6.62	lir::ChebyshevI::HighShelfBase Struct Reference	62
6.63	lir::Butterworth::HighShelfBase Struct Reference	63
6.64	lir::RBJ::IIRNotch Struct Reference	63

6.64.1 Detailed Description	63
6.64.2 Member Function Documentation	63
6.65 <code>lir::Layout< MaxPoles ></code> Class Template Reference	64
6.65.1 Detailed Description	64
6.66 <code>lir::LayoutBase</code> Class Reference	64
6.66.1 Detailed Description	65
6.67 <code>lir::RBJ::LowPass</code> Struct Reference	65
6.67.1 Detailed Description	66
6.67.2 Member Function Documentation	66
6.68 <code>lir::ChebyshevI::LowPass< FilterOrder, StateType ></code> Struct Template Reference	66
6.68.1 Detailed Description	66
6.68.2 Member Function Documentation	67
6.69 <code>lir::Bessel::LowPass< FilterOrder, StateType ></code> Struct Template Reference	67
6.69.1 Detailed Description	68
6.69.2 Member Function Documentation	69
6.70 <code>lir::Butterworth::LowPass< FilterOrder, StateType ></code> Struct Template Reference	69
6.70.1 Detailed Description	69
6.70.2 Member Function Documentation	70
6.71 <code>lir::ChebyshevII::LowPass< FilterOrder, StateType ></code> Struct Template Reference	70
6.71.1 Detailed Description	70
6.71.2 Member Function Documentation	71
6.72 <code>lir::ChebyshevI::LowPassBase</code> Struct Reference	71
6.73 <code>lir::Bessel::LowPassBase</code> Struct Reference	72
6.74 <code>lir::ChebyshevII::LowPassBase</code> Struct Reference	72
6.75 <code>lir::Butterworth::LowPassBase</code> Struct Reference	72
6.76 <code>lir::LowPassTransform</code> Class Reference	73
6.76.1 Detailed Description	73
6.77 <code>lir::ChebyshevII::LowShelf< FilterOrder, StateType ></code> Struct Template Reference	73
6.77.1 Detailed Description	73
6.77.2 Member Function Documentation	74

6.78	lir::Butterworth::LowShelf< FilterOrder, StateType > Struct Template Reference	74
6.78.1	Detailed Description	75
6.78.2	Member Function Documentation	75
6.79	lir::RBJ::LowShelf Struct Reference	75
6.80	lir::ChebyshevI::LowShelf< FilterOrder, StateType > Struct Template Reference	76
6.80.1	Detailed Description	76
6.80.2	Member Function Documentation	76
6.81	lir::Bessel::LowShelfBase Struct Reference	77
6.82	lir::ChebyshevII::LowShelfBase Struct Reference	77
6.83	lir::Butterworth::LowShelfBase Struct Reference	78
6.84	lir::ChebyshevI::LowShelfBase Struct Reference	78
6.85	lir::Custom::OnePole Struct Reference	78
6.85.1	Detailed Description	79
6.86	lir::PoleFilter< BaseClass, StateType, MaxAnalogPoles, MaxDigitalPoles > Struct Template Reference	79
6.86.1	Detailed Description	79
6.87	lir::PoleFilterBase< AnalogPrototype > Class Template Reference	79
6.87.1	Detailed Description	80
6.88	lir::PoleFilterBase2 Class Reference	80
6.88.1	Detailed Description	80
6.89	lir::PoleZeroPair Struct Reference	81
6.89.1	Detailed Description	81
6.90	lir::RBJ::RBJbase Struct Reference	81
6.90.1	Detailed Description	82
6.91	lir::RootFinder< maxdegree > Struct Template Reference	82
6.92	lir::RootFinderBase Class Reference	82
6.92.1	Detailed Description	83
6.93	lir::SlopeDetector< Channels, Value > Class Template Reference	83
6.94	lir::Custom::SOSCascade< NSOS, StateType > Struct Template Reference	83
6.94.1	Detailed Description	83
6.94.2	Member Function Documentation	84
6.95	lir::TransposedDirectFormII Class Reference	84
6.96	lir::Custom::TwoPole Struct Reference	84
6.96.1	Detailed Description	84
6.97	lir::Bessel::Workspace< FilterOrder > Struct Template Reference	85
6.98	lir::Bessel::WorkspaceBase Struct Reference	85

1 IIR1 -- Realtime C++ filter library

An infinite impulse response (IIR) filter library for Linux, Mac OSX and Windows which implements Bessel, Butterworth, RBJ, Chebychev filters and can easily import coefficients generated by Python (scipy).

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);
```

5. Designing filters with Python's scipy.signal – [Custom.h](#)

```
#####
# Python
# See "elliptic_design.py" for the complete code.
from scipy import signal
order = 4
sos = signal.ellip(order, 5, 40, 0.2, 'low', output='sos')
print(sos) # copy/paste the coefficients over & replace [] with {}

/////
// C++
// part of "iirdemo.cpp"
Iir::Custom::SOSCascade<2> cust;
const double coeff[][6] = {
    {1.665623674062209972e-02,
     -3.924801366970616552e-03,
     1.665623674062210319e-02,
     1.000000000000000000e+00,
     -1.715403014004022175e+00,
     8.100474793174089472e-01},
    {1.000000000000000000e+00,
     -1.369778997100624895e+00,
     1.000000000000000022e+00,
     1.000000000000000000e+00,
     -1.605878925999785656e+00,
     9.538657786383895054e-01}
};
cust.setup(coeff);
```

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/usbdux
```

It's available for 32,64 bit PC and 32,64 bit ARM (Raspberry PI etc). The documentation and the example programs are in:

```
/usr/share/doc/iirl-dev/
```

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 responses have been 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	13
lir::Bessel	16
lir::Butterworth	16
lir::ChebyshevI	17
lir::ChebyshevII	17
lir::RBJ	18

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, FilterOrder, FilterOrder *2 >	79
lir::Bessel::BandPass< FilterOrder, StateType >	24
lir::Butterworth::BandPass< FilterOrder, StateType >	22

lir::ChebyshevI::BandPass< FilterOrder, StateType >	21
lir::ChebyshevII::BandPass< FilterOrder, StateType >	25
lir::BandPassTransform	29
BandShelfBase	
lir::PoleFilter< BandShelfBase, StateType, FilterOrder, FilterOrder *2 >	79
lir::Butterworth::BandShelf< FilterOrder, StateType >	32
lir::ChebyshevI::BandShelf< FilterOrder, StateType >	31
lir::ChebyshevII::BandShelf< FilterOrder, StateType >	30
BandStopBase	
lir::PoleFilter< BandStopBase, StateType, FilterOrder, FilterOrder *2 >	79
lir::Bessel::BandStop< FilterOrder, StateType >	37
lir::Butterworth::BandStop< FilterOrder, StateType >	38
lir::ChebyshevI::BandStop< FilterOrder, StateType >	36
lir::ChebyshevII::BandStop< FilterOrder, StateType >	39
lir::BandStopTransform	42
lir::Biquad	42
lir::Custom::OnePole	78
lir::Custom::TwoPole	84
lir::RBJ::RBJbase	81
lir::RBJ::AllPass	18
lir::RBJ::BandPass1	26
lir::RBJ::BandPass2	27
lir::RBJ::BandShelf	31
lir::RBJ::BandStop	35
lir::RBJ::HighPass	51
lir::RBJ::HighShelf	60
lir::RBJ::IIRNotch	63
lir::RBJ::LowPass	65
lir::RBJ::LowShelf	75
lir::Cascade	46
lir::PoleFilterBase2	80
lir::PoleFilterBase< AnalogPrototype >	79
lir::PoleFilterBase< AnalogLowPass >	79

lir::Bessel::BandPassBase	29
lir::Bessel::BandStopBase	40
lir::Bessel::HighPassBase	57
lir::Bessel::LowPassBase	72
lir::Butterworth::BandPassBase	28
lir::Butterworth::BandStopBase	41
lir::Butterworth::HighPassBase	56
lir::Butterworth::LowPassBase	72
lir::ChebyshevI::BandPassBase	28
lir::ChebyshevI::BandStopBase	40
lir::ChebyshevI::HighPassBase	57
lir::ChebyshevI::LowPassBase	71
lir::ChebyshevII::BandPassBase	29
lir::ChebyshevII::BandStopBase	41
lir::ChebyshevII::HighPassBase	57
lir::ChebyshevII::LowPassBase	72
lir::PoleFilterBase< AnalogLowShelf >	79
lir::Bessel::LowShelfBase	77
lir::Butterworth::BandShelfBase	34
lir::Butterworth::HighShelfBase	63
lir::Butterworth::LowShelfBase	78
lir::ChebyshevI::BandShelfBase	33
lir::ChebyshevI::HighShelfBase	62
lir::ChebyshevI::LowShelfBase	78
lir::ChebyshevII::BandShelfBase	34
lir::ChebyshevII::HighShelfBase	62
lir::ChebyshevII::LowShelfBase	77
lir::CascadeStages< MaxStages, StateType >	47
lir::CascadeStages< NSOS, StateType >	47
lir::Custom::SOSCascade< NSOS, StateType >	83
lir::CascadeStages<(MaxAnalogPoles+1)/2, StateType >	47
lir::PoleFilter< HighPassBase, StateType, FilterOrder >	79

lir::Bessel::HighPass< FilterOrder, StateType >	52
lir::Butterworth::HighPass< FilterOrder, StateType >	54
lir::ChebyshevI::HighPass< FilterOrder, StateType >	53
lir::ChebyshevII::HighPass< FilterOrder, StateType >	55
lir::PoleFilter< HighShelfBase, StateType, FilterOrder >	79
lir::Butterworth::HighShelf< FilterOrder, StateType >	59
lir::ChebyshevI::HighShelf< FilterOrder, StateType >	61
lir::ChebyshevII::HighShelf< FilterOrder, StateType >	58
lir::PoleFilter< LowPassBase, StateType, FilterOrder >	79
lir::Bessel::LowPass< FilterOrder, StateType >	67
lir::Butterworth::LowPass< FilterOrder, StateType >	69
lir::ChebyshevI::LowPass< FilterOrder, StateType >	66
lir::ChebyshevII::LowPass< FilterOrder, StateType >	70
lir::PoleFilter< LowShelfBase, StateType, FilterOrder >	79
lir::Butterworth::LowShelf< FilterOrder, StateType >	74
lir::ChebyshevI::LowShelf< FilterOrder, StateType >	76
lir::ChebyshevII::LowShelf< FilterOrder, StateType >	73
lir::CascadeStages<(MaxDigitalPoles+1)/2, StateType >	47
lir::PoleFilter< BaseClass, StateType, MaxAnalogPoles, MaxDigitalPoles >	79
lir::PoleFilter< BandPassBase, StateType, FilterOrder, FilterOrder *2 >	79
lir::PoleFilter< BandShelfBase, StateType, FilterOrder, FilterOrder *2 >	79
lir::PoleFilter< BandStopBase, StateType, FilterOrder, FilterOrder *2 >	79
complex_pair_t	
lir::ComplexPair	49
lir::DirectFormI	50
lir::DirectFormII	50
lir::EnvelopeFollower< Channels, Value >	50
HighPassBase	
lir::PoleFilter< HighPassBase, StateType, FilterOrder >	79
lir::HighPassTransform	58
HighShelfBase	
lir::PoleFilter< HighShelfBase, StateType, FilterOrder >	79
lir::Layout< MaxPoles >	64

lir::Layout< MaxAnalogPoles >	64
lir::Layout< MaxDigitalPoles >	64
lir::LayoutBase	64
lir::Bessel::AnalogLowPass	19
lir::Bessel::AnalogLowShelf	20
lir::Butterworth::AnalogLowPass	19
lir::Butterworth::AnalogLowShelf	20
lir::ChebyshevI::AnalogLowPass	19
lir::ChebyshevI::AnalogLowShelf	21
lir::ChebyshevII::AnalogLowPass	20
lir::ChebyshevII::AnalogLowShelf	21
LowPassBase	
lir::PoleFilter< LowPassBase, StateType, FilterOrder >	79
lir::LowPassTransform	73
LowShelfBase	
lir::PoleFilter< LowShelfBase, StateType, FilterOrder >	79
lir::PoleZeroPair	81
lir::BiquadPoleState	46
lir::RootFinderBase	82
lir::RootFinder< maxdegree >	82
lir::RootFinder< FilterOrder >	82
lir::SlopeDetector< Channels, Value >	83
lir::TransposedDirectFormII	84
lir::Bessel::WorkspaceBase	85
lir::Bessel::Workspace< FilterOrder >	85
BaseClass	
lir::PoleFilter< BaseClass, StateType, MaxAnalogPoles, MaxDigitalPoles >	79

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	19
lir::ChebyshevI::AnalogLowPass	19
lir::Butterworth::AnalogLowPass	19
lir::ChebyshevII::AnalogLowPass	20
lir::Bessel::AnalogLowShelf	20
lir::Butterworth::AnalogLowShelf	20
lir::ChebyshevI::AnalogLowShelf	21
lir::ChebyshevII::AnalogLowShelf	21
lir::RootFinderBase::Array	21
lir::ChebyshevI::BandPass< FilterOrder, StateType >	21
lir::Butterworth::BandPass< FilterOrder, StateType >	22
lir::Bessel::BandPass< FilterOrder, StateType >	24
lir::ChebyshevII::BandPass< FilterOrder, StateType >	25
lir::RBJ::BandPass1	26
lir::RBJ::BandPass2	27
lir::ChebyshevI::BandPassBase	28
lir::Butterworth::BandPassBase	28
lir::Bessel::BandPassBase	29
lir::ChebyshevII::BandPassBase	29
lir::BandPassTransform	29
lir::ChebyshevII::BandShelf< FilterOrder, StateType >	30
lir::RBJ::BandShelf	31
lir::ChebyshevI::BandShelf< FilterOrder, StateType >	31
lir::Butterworth::BandShelf< FilterOrder, StateType >	32
lir::ChebyshevI::BandShelfBase	33
lir::Butterworth::BandShelfBase	34
lir::ChebyshevII::BandShelfBase	34
lir::RBJ::BandStop	35
lir::ChebyshevI::BandStop< FilterOrder, StateType >	36
lir::Bessel::BandStop< FilterOrder, StateType >	37
lir::Butterworth::BandStop< FilterOrder, StateType >	38
lir::ChebyshevII::BandStop< FilterOrder, StateType >	39

lir::Bessel::BandStopBase	40
lir::ChebyshevI::BandStopBase	40
lir::Butterworth::BandStopBase	41
lir::ChebyshevII::BandStopBase	41
lir::BandStopTransform	42
lir::Biquad	42
lir::BiquadPoleState	46
lir::Cascade	46
lir::CascadeStages< MaxStages, StateType >	47
lir::ComplexPair	49
lir::DirectFormI	50
lir::DirectFormII	50
lir::EnvelopeFollower< Channels, Value >	50
lir::RBJ::HighPass	51
lir::Bessel::HighPass< FilterOrder, StateType >	52
lir::ChebyshevI::HighPass< FilterOrder, StateType >	53
lir::Butterworth::HighPass< FilterOrder, StateType >	54
lir::ChebyshevII::HighPass< FilterOrder, StateType >	55
lir::Butterworth::HighPassBase	56
lir::ChebyshevI::HighPassBase	57
lir::Bessel::HighPassBase	57
lir::ChebyshevII::HighPassBase	57
lir::HighPassTransform	58
lir::ChebyshevII::HighShelf< FilterOrder, StateType >	58
lir::Butterworth::HighShelf< FilterOrder, StateType >	59
lir::RBJ::HighShelf	60
lir::ChebyshevI::HighShelf< FilterOrder, StateType >	61
lir::ChebyshevII::HighShelfBase	62
lir::ChebyshevI::HighShelfBase	62
lir::Butterworth::HighShelfBase	63
lir::RBJ::IIRNotch	63
lir::Layout< MaxPoles >	64

lir::LayoutBase	64
lir::RBJ::LowPass	65
lir::ChebyshevI::LowPass< FilterOrder, StateType >	66
lir::Bessel::LowPass< FilterOrder, StateType >	67
lir::Butterworth::LowPass< FilterOrder, StateType >	69
lir::ChebyshevII::LowPass< FilterOrder, StateType >	70
lir::ChebyshevI::LowPassBase	71
lir::Bessel::LowPassBase	72
lir::ChebyshevII::LowPassBase	72
lir::Butterworth::LowPassBase	72
lir::LowPassTransform	73
lir::ChebyshevII::LowShelf< FilterOrder, StateType >	73
lir::Butterworth::LowShelf< FilterOrder, StateType >	74
lir::RBJ::LowShelf	75
lir::ChebyshevI::LowShelf< FilterOrder, StateType >	76
lir::Bessel::LowShelfBase	77
lir::ChebyshevII::LowShelfBase	77
lir::Butterworth::LowShelfBase	78
lir::ChebyshevI::LowShelfBase	78
lir::Custom::OnePole	78
lir::PoleFilter< BaseClass, StateType, MaxAnalogPoles, MaxDigitalPoles >	79
lir::PoleFilterBase< AnalogPrototype >	79
lir::PoleFilterBase2	80
lir::PoleZeroPair	81
lir::RBJ::RBJbase	81
lir::RootFinder< maxdegree >	82
lir::RootFinderBase	82
lir::SlopeDetector< Channels, Value >	83
lir::Custom::SOSCascade< NSOS, StateType >	83
lir::TransposedDirectFormII	84
lir::Custom::TwoPole	84
lir::Bessel::Workspace< FilterOrder >	85

[lir::Bessel::WorkspaceBase](#)

85

5 Namespace Documentation

5.1 lir Namespace Reference

Namespaces

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

Classes

- class [BandPassTransform](#)
- class [BandStopTransform](#)
- class [Biquad](#)
- 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.txt 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 lir::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 Iir::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:

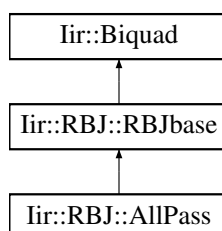
<http://www.musicdsp.org/files/Audio-EQ-Cookbook.txt>

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:



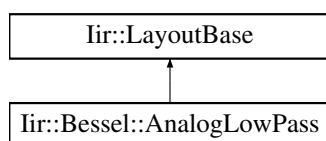
Additional Inherited Members

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:

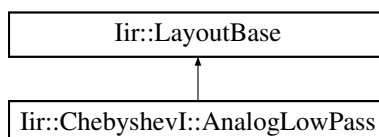


The documentation for this class was generated from the following files:

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

6.3 Iir::ChebyshevI::AnalogLowPass Class Reference

Inheritance diagram for Iir::ChebyshevI::AnalogLowPass:



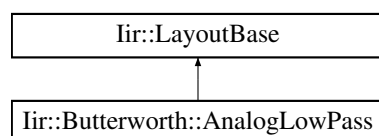
The documentation for this class was generated from the following files:

- 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

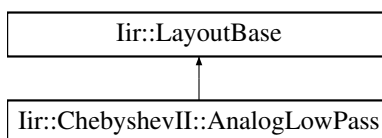
Half-band analog prototypes (s-plane)

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:

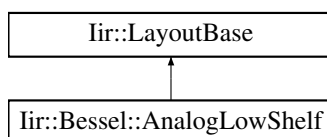


The documentation for this class was generated from the following files:

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

6.6 Iir::Bessel::AnalogLowShelf Class Reference

Inheritance diagram for Iir::Bessel::AnalogLowShelf:

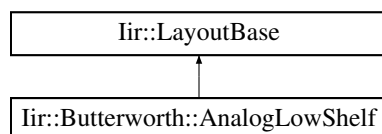


The documentation for this class was generated from the following files:

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

6.7 Iir::Butterworth::AnalogLowShelf Class Reference

Inheritance diagram for Iir::Butterworth::AnalogLowShelf:

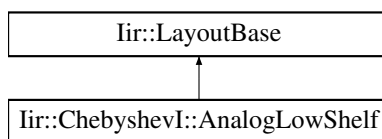


The documentation for this class was generated from the following files:

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

6.8 Iir::ChebyshevI::AnalogLowShelf Class Reference

Inheritance diagram for Iir::ChebyshevI::AnalogLowShelf:

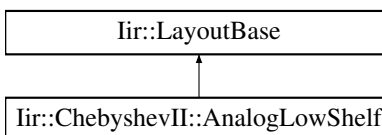


The documentation for this class was generated from the following files:

- 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

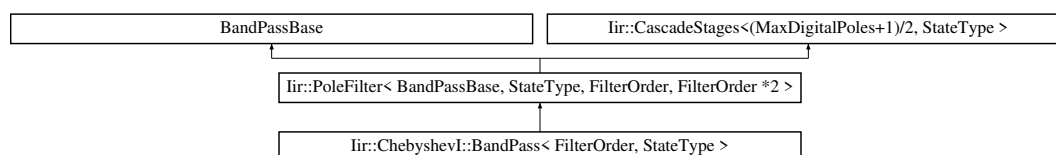
The documentation for this struct was generated from the following file:

- iir/RootFinder.h

6.11 Iir::ChebyshevI::BandPass< FilterOrder, StateType > Struct Template Reference

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

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



Public Member Functions

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

6.11.1 Detailed Description

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

[ChebyshevI](#) bandpass filter

Parameters

<i>FilterOrder</i>	The order of the filter.
<i>StateType</i>	The filter topology: DirectFormI , DirectFormII , ...

6.11.2 Member Function Documentation

6.11.2.1 setup()

```
template<int FilterOrder, class StateType = DEFAULT_STATE>
void Iir::ChebyshevI::BandPass< FilterOrder, 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

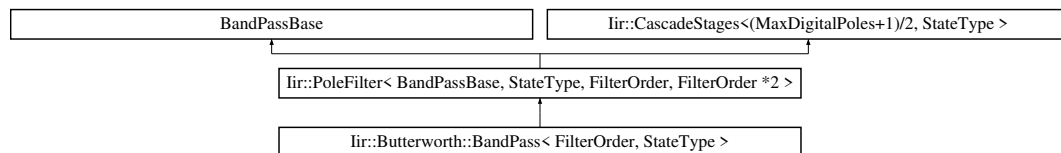
The documentation for this struct was generated from the following file:

- [iir/ChebyshevI.h](#)

6.12 Iir::Butterworth::BandPass< FilterOrder, StateType > Struct Template Reference

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

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



Public Member Functions

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

6.12.1 Detailed Description

```
template<int FilterOrder, class StateType = DEFAULT_STATE>
struct lir::Butterworth::BandPass< FilterOrder, StateType >
```

[Butterworth](#) Bandpass filter.

Parameters

<i>FilterOrder</i>	The order of the filter.
<i>StateType</i>	The filter topology: DirectFormI , DirectFormII , ...

6.12.2 Member Function Documentation

6.12.2.1 `setup()`

```
template<int FilterOrder, class StateType = DEFAULT_STATE>
void Iir::Butterworth::BandPass< FilterOrder, 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

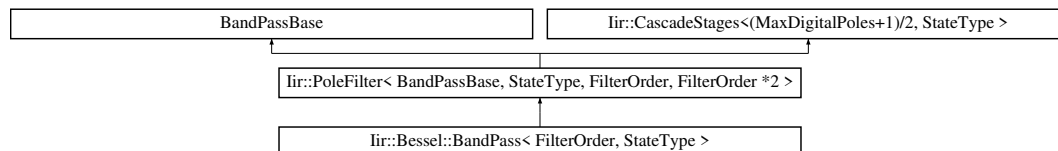
The documentation for this struct was generated from the following file:

- `iir/Butterworth.h`

6.13 `lir::Bessel::BandPass< FilterOrder, StateType >` Struct Template Reference

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

Inheritance diagram for `lir::Bessel::BandPass< FilterOrder, StateType >`:



Public Member Functions

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

6.13.1 Detailed Description

```
template<int FilterOrder, class StateType = DEFAULT_STATE>
struct lir::Bessel::BandPass< FilterOrder, StateType >
```

[Bessel](#) bandpass.

Parameters

<i>FilterOrder</i>	The order of the filter.
<i>StateType</i>	The filter topology: DirectFormI , DirectFormII , ...

6.13.2 Member Function Documentation

6.13.2.1 `setup()`

```
template<int FilterOrder, class StateType = DEFAULT_STATE>
void lir::Bessel::BandPass< FilterOrder, 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

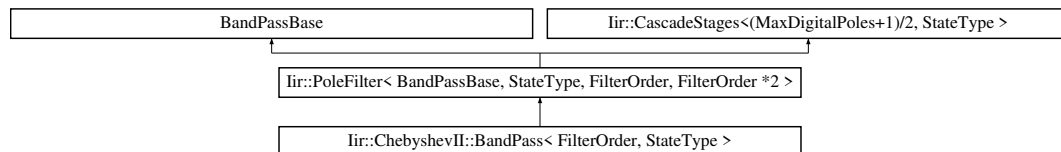
The documentation for this struct was generated from the following file:

- iir/Bessel.h

6.14 Iir::ChebyshevII::BandPass< FilterOrder, StateType > Struct Template Reference

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

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



Public Member Functions

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

6.14.1 Detailed Description

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

[ChebyshevII](#) bandpass filter

Parameters

<i>FilterOrder</i>	The order of the filter.
<i>StateType</i>	The filter topology: DirectFormI , DirectFormII , ...

6.14.2 Member Function Documentation

6.14.2.1 setup()

```
template<int FilterOrder, class StateType = DEFAULT_STATE>
void Iir::ChebyshevII::BandPass< FilterOrder, 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

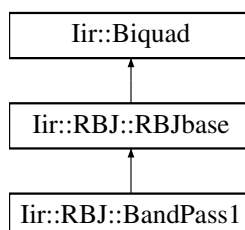
The documentation for this struct was generated from the following file:

- `iir/ChebyshevII.h`

6.15 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

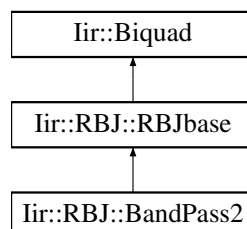
The documentation for this struct was generated from the following files:

- 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

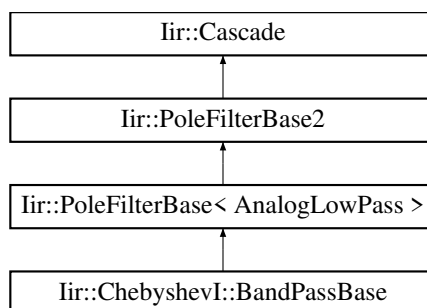
<i>sampleRate</i>	Sampling rate
<i>centerFrequency</i>	Center frequency of the bandpass
<i>bandWidth</i>	Bandwidth of the bandpass

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

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

6.17 Iir::ChebyshevI::BandPassBase Struct Reference

Inheritance diagram for Iir::ChebyshevI::BandPassBase:

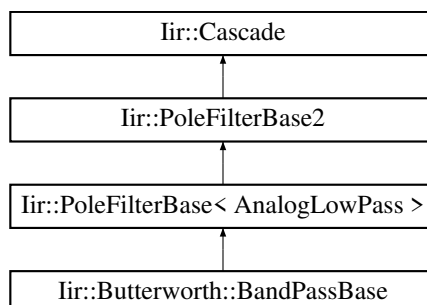


The documentation for this struct was generated from the following files:

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

6.18 Iir::Butterworth::BandPassBase Struct Reference

Inheritance diagram for Iir::Butterworth::BandPassBase:

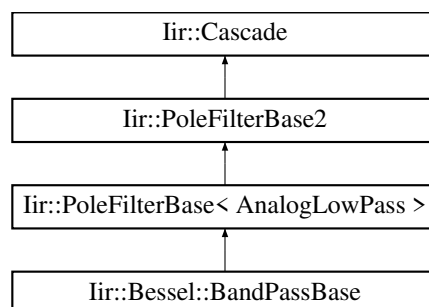


The documentation for this struct was generated from the following files:

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

6.19 Iir::Bessel::BandPassBase Struct Reference

Inheritance diagram for Iir::Bessel::BandPassBase:

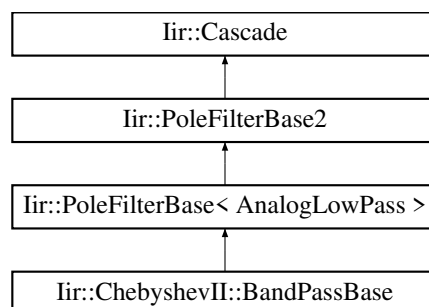


The documentation for this struct was generated from the following files:

- 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

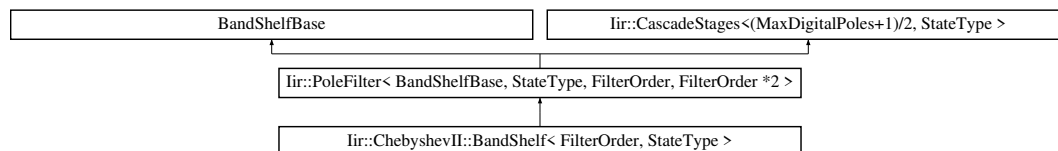
The documentation for this class was generated from the following files:

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

6.22 `lir::ChebyshevII::BandShelf< FilterOrder, StateType >` Struct Template Reference

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

Inheritance diagram for `lir::ChebyshevII::BandShelf< FilterOrder, StateType >`:



Public Member Functions

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

6.22.1 Detailed Description

```
template<int FilterOrder, class StateType = DEFAULT_STATE>
struct lir::ChebyshevII::BandShelf< FilterOrder, StateType >
```

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

Parameters

<i>FilterOrder</i>	The order of the filter.
<i>StateType</i>	The filter topology: DirectFormI , DirectFormII , ...

6.22.2 Member Function Documentation

6.22.2.1 `setup()`

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

Calculates the coefficients of the filter

Parameters

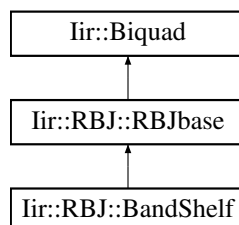
<i>sampleRate</i>	Sampling rate
<i>centerFrequency</i>	Center frequency of the bandpass
<i>widthFrequency</i>	Width of the bandpass
<i>gainDb</i>	Gain in the passband. The stopband has always 0dB.
<i>stopBandDb</i>	Permitted ripples in dB in the stopband

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

- iir/ChebyshevI.h

6.23 Iir::RBJ::BandShelf Struct Reference

Inheritance diagram for Iir::RBJ::BandShelf:



Additional Inherited Members

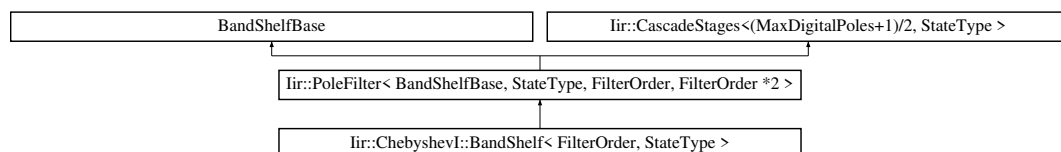
The documentation for this struct was generated from the following files:

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

6.24 Iir::ChebyshevI::BandShelf< FilterOrder, StateType > Struct Template Reference

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

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



Public Member Functions

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

6.24.1 Detailed Description

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

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

Parameters

<i>FilterOrder</i>	The order of the filter.
<i>StateType</i>	The filter topology: DirectFormI , DirectFormII , ...

6.24.2 Member Function Documentation

6.24.2.1 setup()

```
template<int FilterOrder, class StateType = DEFAULT_STATE>
void Iir::ChebyshevI::BandShelf< FilterOrder, 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.

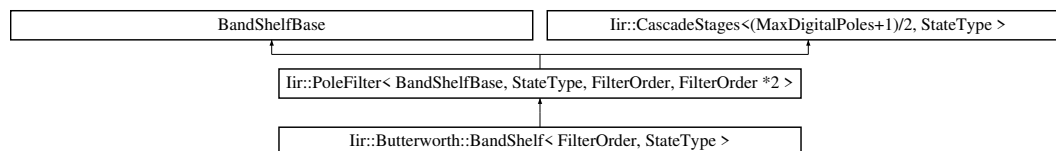
The documentation for this struct was generated from the following file:

- [iir/ChebyshevI.h](#)

6.25 Iir::Butterworth::BandShelf< FilterOrder, StateType > Struct Template Reference

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

Inheritance diagram for Iir::Butterworth::BandShelf< FilterOrder, StateType >:



Public Member Functions

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

6.25.1 Detailed Description

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

[Butterworth](#) Bandshef filter: it is a bandpass filter which amplifies at a specified gain in dB the frequencies in the passband.

Parameters

<i>FilterOrder</i>	The order of the filter.
<i>StateType</i>	The filter topology: DirectFormI , DirectFormII , ...

6.25.2 Member Function Documentation

6.25.2.1 setup()

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

Calculates the coefficients

Parameters

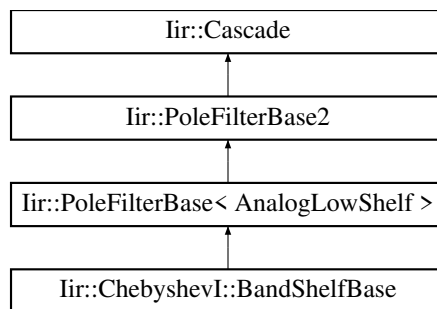
<i>sampleRate</i>	Sampling rate
<i>centerFrequency</i>	Centre frequency of the passband
<i>widthFrequency</i>	Width of the passband
<i>gainDb</i>	The gain in the passband

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

- `iir/Butterworth.h`

6.26 Iir::ChebyshevI::BandShelfBase Struct Reference

Inheritance diagram for Iir::ChebyshevI::BandShelfBase:

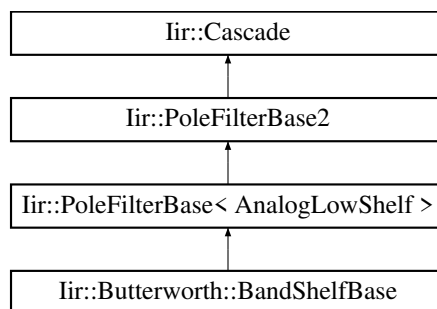


The documentation for this struct was generated from the following files:

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

6.27 Iir::Butterworth::BandShelfBase Struct Reference

Inheritance diagram for `Iir::Butterworth::BandShelfBase`:

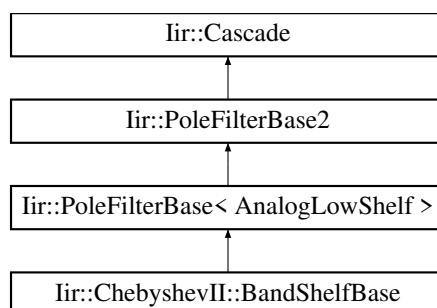


The documentation for this struct was generated from the following files:

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

6.28 Iir::ChebyshevII::BandShelfBase Struct Reference

Inheritance diagram for `Iir::ChebyshevII::BandShelfBase`:



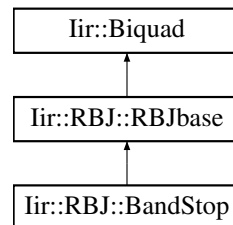
The documentation for this struct was generated from the following files:

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

6.29 Iir::RBJ::BandStop Struct Reference

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

Inheritance diagram for Iir::RBJ::BandStop:



Public Member Functions

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

6.29.1 Detailed Description

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

6.29.2 Member Function Documentation

6.29.2.1 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

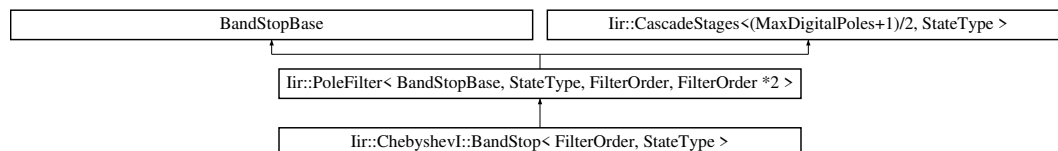
The documentation for this struct was generated from the following files:

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

6.30 `lir::ChebyshevI::BandStop< FilterOrder, StateType >` Struct Template Reference

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

Inheritance diagram for `lir::ChebyshevI::BandStop< FilterOrder, StateType >`:



Public Member Functions

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

6.30.1 Detailed Description

```
template<int FilterOrder, class StateType = DEFAULT_STATE>
struct lir::ChebyshevI::BandStop< FilterOrder, StateType >
```

[ChebyshevI](#) bandstop filter

Parameters

<i>FilterOrder</i>	The order of the filter.
<i>StateType</i>	The filter topology: DirectFormI , DirectFormII , ...

6.30.2 Member Function Documentation

6.30.2.1 `setup()`

```
template<int FilterOrder, class StateType = DEFAULT_STATE>
void lir::ChebyshevI::BandStop< FilterOrder, 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 width of the notch
<i>rippleDb</i>	Permitted ripples in dB in the passband

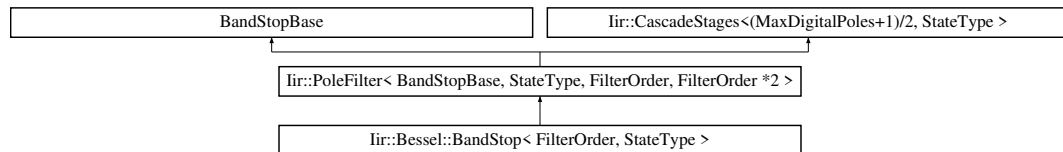
The documentation for this struct was generated from the following file:

- iir/ChebyshevI.h

6.31 Iir::Bessel::BandStop< FilterOrder, StateType > Struct Template Reference

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

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



Public Member Functions

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

6.31.1 Detailed Description

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

[Bessel](#) bandstop.

Parameters

<i>FilterOrder</i>	The order of the filter.
<i>StateType</i>	The filter topology: DirectFormI , DirectFormII , ...

6.31.2 Member Function Documentation

6.31.2.1 setup()

```
template<int FilterOrder, class StateType = DEFAULT_STATE>
void Iir::Bessel::BandStop< FilterOrder, 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

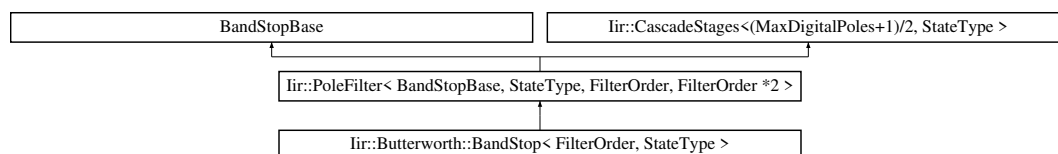
The documentation for this struct was generated from the following file:

- `iir/Bessel.h`

6.32 `Iir::Butterworth::BandStop< FilterOrder, StateType >` Struct Template Reference

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

Inheritance diagram for `Iir::Butterworth::BandStop< FilterOrder, StateType >`:



Public Member Functions

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

6.32.1 Detailed Description

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

[Butterworth](#) Bandstop filter.

Parameters

<i>FilterOrder</i>	The order of the filter.
<i>StateType</i>	The filter topology: DirectFormI , DirectFormII , ...

6.32.2 Member Function Documentation

6.32.2.1 `setup()`

```
template<int FilterOrder, class StateType = DEFAULT_STATE>
void Iir::Butterworth::BandStop< FilterOrder, 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

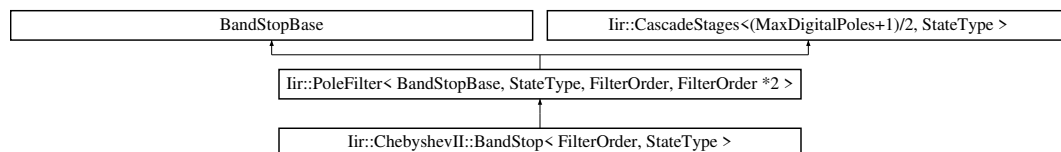
The documentation for this struct was generated from the following file:

- iir/Butterworth.h

6.33 Iir::ChebyshevII::BandStop< FilterOrder, StateType > Struct Template Reference

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

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



Public Member Functions

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

6.33.1 Detailed Description

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

[ChebyshevII](#) bandstop filter.

Parameters

<i>FilterOrder</i>	The order of the filter.
<i>StateType</i>	The filter topology: DirectFormI , DirectFormII , ...

6.33.2 Member Function Documentation

6.33.2.1 setup()

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

Calculates the coefficients of the filter

Parameters

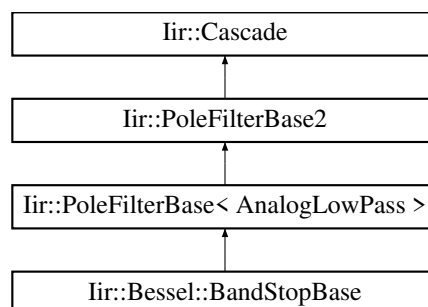
<i>sampleRate</i>	Sampling rate
<i>centerFrequency</i>	Center frequency of the bandstop
<i>widthFrequency</i>	Width of the bandstop
<i>stopBandDb</i>	Permitted ripples in dB in the stopband

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

- iir/ChebyshevII.h

6.34 Iir::Bessel::BandStopBase Struct Reference

Inheritance diagram for Iir::Bessel::BandStopBase:

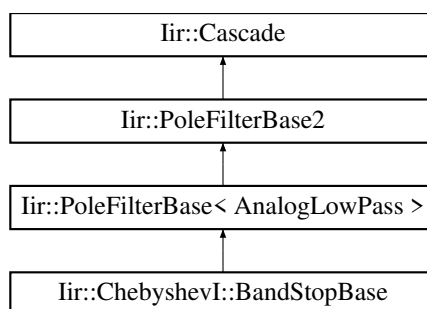


The documentation for this struct was generated from the following files:

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

6.35 Iir::ChebyshevI::BandStopBase Struct Reference

Inheritance diagram for Iir::ChebyshevI::BandStopBase:

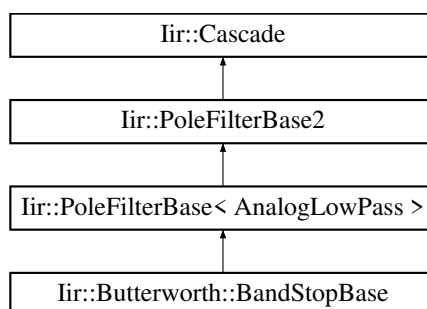


The documentation for this struct was generated from the following files:

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

6.36 Iir::Butterworth::BandStopBase Struct Reference

Inheritance diagram for `Iir::Butterworth::BandStopBase`:

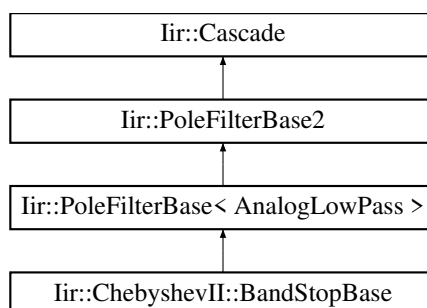


The documentation for this struct was generated from the following files:

- `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

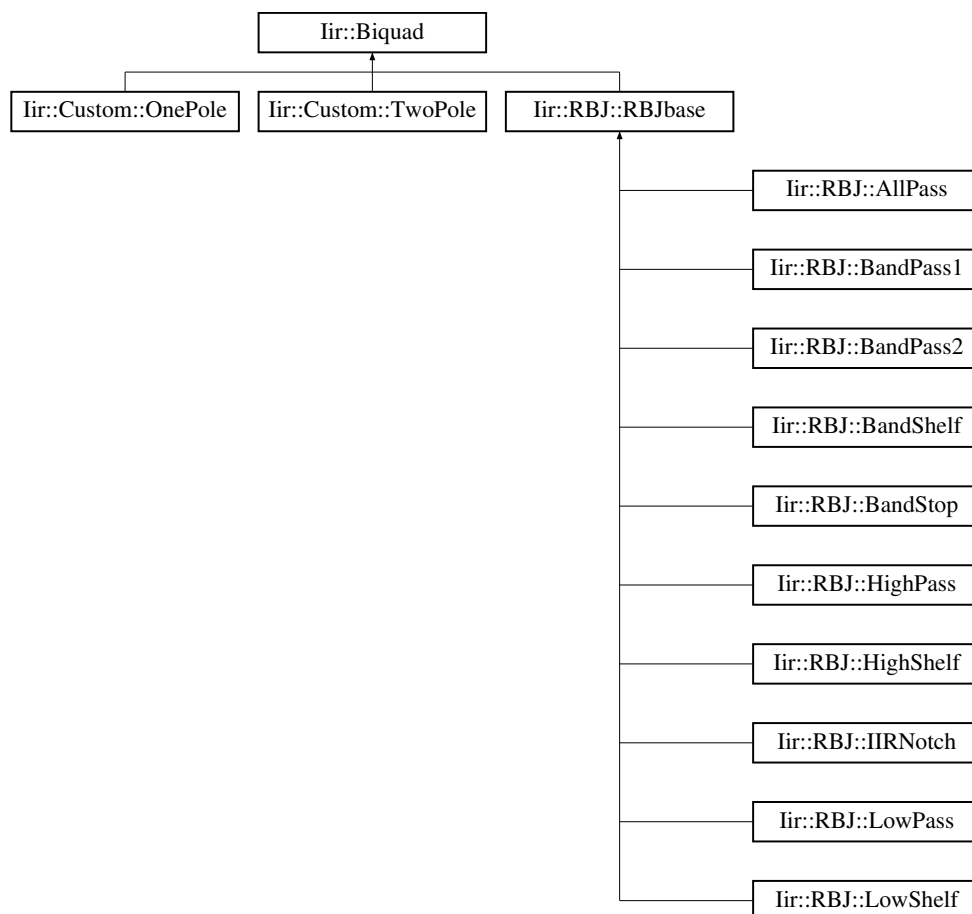
low pass to band stop transform

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:



Public Member Functions

- `complex_t response` (double normalizedFrequency) const
- `std::vector< PoleZeroPair > getPoleZeros` () const
- `double getA0` () const
- `double getA1` () const
- `double getA2` () const
- `double getB0` () const
- `double getB1` () const
- `double getB2` () const
- `template<class StateType , typename Sample > Sample filter` (Sample s, StateType &state) const
- `void setCoefficients` (double a0, double a1, double a2, double b0, double b1, double b2)
- `void setOnePole` (complex_t pole, complex_t zero)
- `void setTwoPole` (complex_t pole1, complex_t zero1, complex_t pole2, complex_t zero2)
- `void setPoleZeroPair` (const PoleZeroPair &pair)
- `void setIdentity` ()
- `void applyScale` (double scale)

6.39.1 Member Function Documentation

6.39.1.1 applyScale()

```
void Iir::Biquad::applyScale (
    double scale )
```

Performs scaling operation on the FIR coefficients

Parameters

<i>scale</i>	Multplies the coefficients b0,b1,b2 with the scaling factor scale.
--------------	--

6.39.1.2 filter()

```
template<class StateType , typename Sample >
Sample Iir::Biquad::filter (
    Sample s,
    StateType & state ) const [inline]
```

Filter a sample with the coefficients provided here and the State provided as an argument.

Parameters

<i>s</i>	The sample to be filtered.
<i>state</i>	The Delay lines (instance of a state from State.h)

6.39.1.3 getA0()

```
double Iir::Biquad::getA0 ( ) const [inline]
```

Returns 1st IIR coefficient (usually one)

6.39.1.4 getA1()

```
double Iir::Biquad::getA1 ( ) const [inline]
```

Returns 2nd IIR coefficient

6.39.1.5 getA2()

```
double Iir::Biquad::getA2 ( ) const [inline]
```

Returns 3rd IIR coefficient

6.39.1.6 getB0()

```
double Iir::Biquad::getB0 ( ) const [inline]
```

Returns 1st FIR coefficient

6.39.1.7 getB1()

```
double Iir::Biquad::getB1 ( ) const [inline]
```

Returns 2nd FIR coefficient

6.39.1.8 getB2()

```
double Iir::Biquad::getB2 ( ) const [inline]
```

Returns 3rd FIR coefficient

6.39.1.9 getPoleZeros()

```
std::vector< PoleZeroPair > Iir::Biquad::getPoleZeros ( ) const
```

Returns the pole / zero Pairs as a vector

6.39.1.10 response()

```
complex_t Iir::Biquad::response (
    double normalizedFrequency ) const
```

Calculate filter response at the given normalized frequency.

Gets the frequency response of the [Biquad](#)

Parameters

<i>normalizedFrequency</i>	Normalised frequency (0 to 0.5)
----------------------------	---------------------------------

6.39.1.11 setCoefficients()

```
void Iir::Biquad::setCoefficients (
    double a0,
    double a1,
    double a2,
    double b0,
    double b1,
    double b2 )
```

Sets all coefficients

Parameters

<i>a0</i>	1st IIR coefficient
<i>a1</i>	2nd IIR coefficient
<i>a2</i>	3rd IIR coefficient
<i>b0</i>	1st FIR coefficient
<i>b1</i>	2nd FIR coefficient
<i>b2</i>	3rd FIR coefficient

6.39.1.12 setIdentity()

```
void Iir::Biquad::setIdentity ( )
```

Sets the coefficients as pass through. (b0=1,a0=1, rest zero)

6.39.1.13 setOnePole()

```
void Iir::Biquad::setOnePole (
    complex_t pole,
    complex_t zero )
```

Sets one (real) pole and zero. Throws exception if imaginary components.

6.39.1.14 setPoleZeroPair()

```
void Iir::Biquad::setPoleZeroPair (
    const PoleZeroPair & pair ) [inline]
```

Sets a complex conjugate pair

6.39.1.15 setTwoPole()

```
void Iir::Biquad::setTwoPole (
    complex_t pole1,
    complex_t zero1,
    complex_t pole2,
    complex_t zero2 )
```

Sets two poles/zoes as a pair. Needs to be complex conjugate.

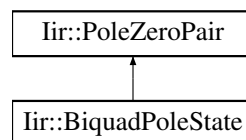
The documentation for this class was generated from the following files:

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

6.40 Iir::BiquadPoleState Struct Reference

```
#include <Biquad.h>
```

Inheritance diagram for Iir::BiquadPoleState:



6.40.1 Detailed Description

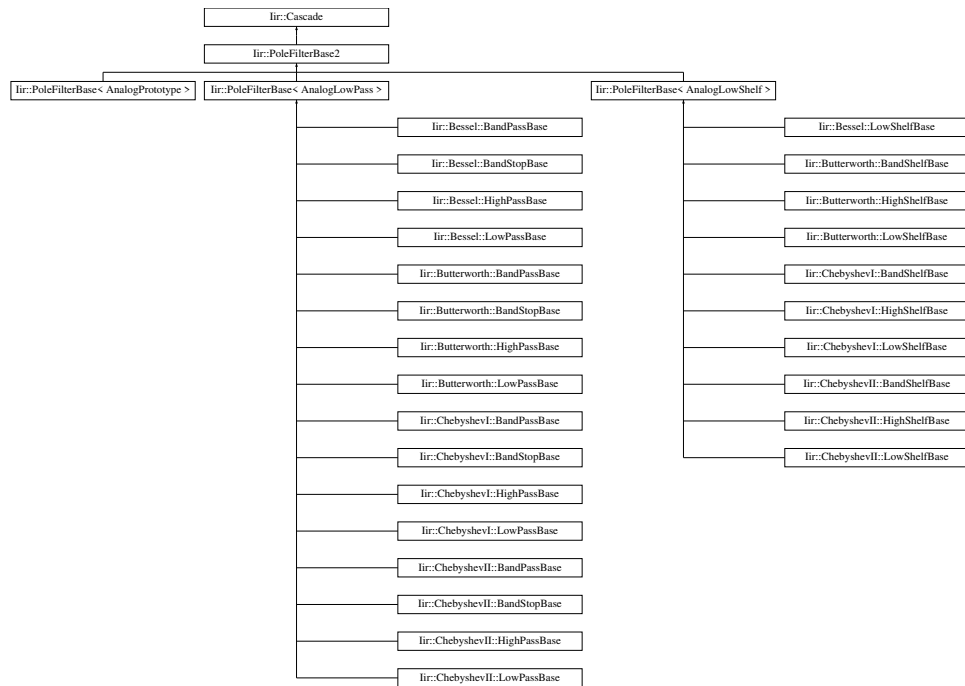
Expresses a biquad as a pair of pole/zeros, with gain values so that the coefficients can be reconstructed precisely.

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

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

6.41 Iir::Cascade Class Reference

Inheritance diagram for Iir::Cascade:



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

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

6.42 `lir::CascadeStages< MaxStages, StateType >` Class Template Reference

```
#include <Cascade.h>
```

Public Member Functions

- `void reset ()`
- `void setup (const double(&sosCoefficients)[MaxStages][6])`
- `template<typename Sample > Sample filter (const Sample in)`

6.42.1 Detailed Description

```
template<int MaxStages, class StateType>
class lir::CascadeStages< MaxStages, StateType >
```

Storage for [Cascade](#): This holds a chain of 2nd order filters with its coefficients.

6.42.2 Member Function Documentation

6.42.2.1 filter()

```
template<int MaxStages, class StateType>
template<typename Sample >
Sample Iir::CascadeStages< MaxStages, StateType >::filter (
    const Sample in ) [inline]
```

Filters one sample through the whole chain of biquads and return the result

Parameters

<i>in</i>	Sample to be filtered
-----------	-----------------------

6.42.2.2 reset()

```
template<int MaxStages, class StateType>
void Iir::CascadeStages< MaxStages, StateType >::reset ( ) [inline]
```

Resets all biquads (i.e. the delay lines but not the coefficients)

6.42.2.3 setup()

```
template<int MaxStages, class StateType>
void Iir::CascadeStages< MaxStages, StateType >::setup (
    const double(&) sosCoefficients[MaxStages][6] ) [inline]
```

Sets the coefficients of the whole chain of biquads.

Parameters

<i>sosCoefficients</i>	2D array in Python style sos ordering: 0-2: FIR, 3-5: IIR coeff.
------------------------	--

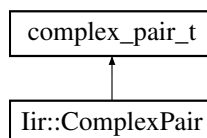
The documentation for this class was generated from the following file:

- iir/Cascade.h

6.43 Iir::ComplexPair Struct Reference

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

Inheritance diagram for Iir::ComplexPair:



6.43.1 Detailed Description

A conjugate or real pair

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

- iir/Types.h

6.44 `iir::DirectFormI` Class Reference

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

6.44.1 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`

6.45 `iir::DirectFormII` Class Reference

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

6.45.1 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`

6.46 `iir::EnvelopeFollower< Channels, Value >` Class Template Reference

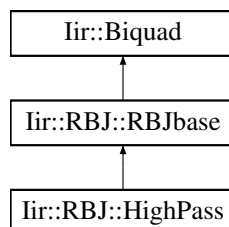
The documentation for this class was generated from the following file:

- `iir/Utilities.h`

6.47 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)

6.47.1 Detailed Description

Highpass.

6.47.2 Member Function Documentation

6.47.2.1 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.

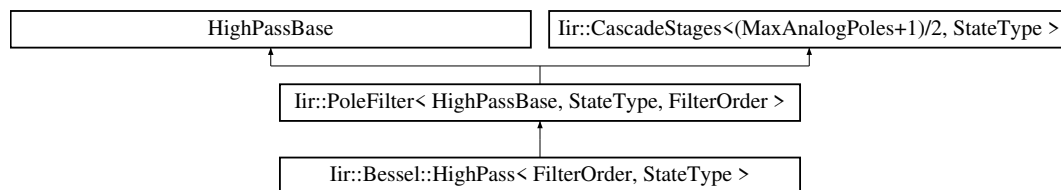
The documentation for this struct was generated from the following files:

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

6.48 Iir::Bessel::HighPass< FilterOrder, StateType > Struct Template Reference

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

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



Public Member Functions

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

6.48.1 Detailed Description

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

[Bessel](#) Highpass.

Parameters

<i>FilterOrder</i>	The order of the filter.
<i>StateType</i>	The filter topology: DirectFormI , DirectFormII , ...

6.48.2 Member Function Documentation

6.48.2.1 setup()

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

Calculate the coefficients

Parameters

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

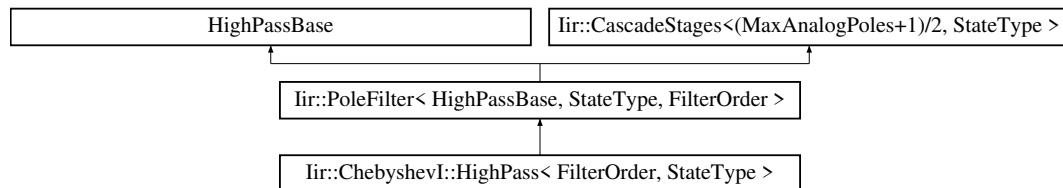
The documentation for this struct was generated from the following file:

- `iir/Bessel.h`

6.49 `lir::ChebyshevI::HighPass< FilterOrder, StateType >` Struct Template Reference

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

Inheritance diagram for `lir::ChebyshevI::HighPass< FilterOrder, StateType >`:



Public Member Functions

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

6.49.1 Detailed Description

```
template<int FilterOrder, class StateType = DEFAULT_STATE>
struct lir::ChebyshevI::HighPass< FilterOrder, StateType >
```

[ChebyshevI](#) highpass filter

Parameters

<i>FilterOrder</i>	The order of the filter.
<i>StateType</i>	The filter topology: DirectFormI , DirectFormII , ...

6.49.2 Member Function Documentation

6.49.2.1 `setup()`

```
template<int FilterOrder, class StateType = DEFAULT_STATE>
void lir::ChebyshevI::HighPass< FilterOrder, 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

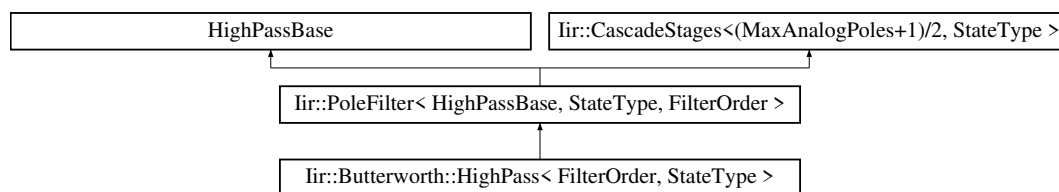
The documentation for this struct was generated from the following file:

- `iir/ChebyshevI.h`

6.50 `lir::Butterworth::HighPass< FilterOrder, StateType >` Struct Template Reference

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

Inheritance diagram for `lir::Butterworth::HighPass< FilterOrder, StateType >`:



Public Member Functions

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

6.50.1 Detailed Description

```
template<int FilterOrder, class StateType = DEFAULT_STATE>
struct lir::Butterworth::HighPass< FilterOrder, StateType >
```

[Butterworth](#) Highpass filter.

Parameters

<i>FilterOrder</i>	The order of the filter.
<i>StateType</i>	The filter topology: DirectFormI , DirectFormII , ...

6.50.2 Member Function Documentation

6.50.2.1 `setup()`

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

Calculates the coefficients

Parameters

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

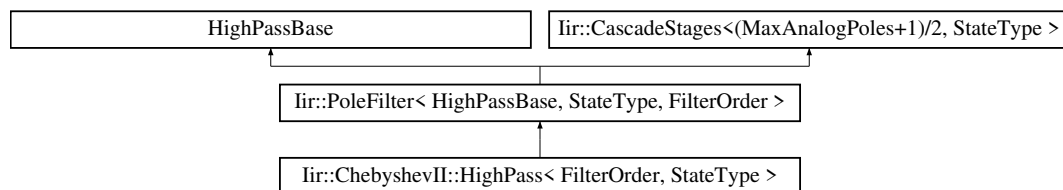
The documentation for this struct was generated from the following file:

- `iir/Butterworth.h`

6.51 `lir::ChebyshevII::HighPass< FilterOrder, StateType >` Struct Template Reference

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

Inheritance diagram for `lir::ChebyshevII::HighPass< FilterOrder, StateType >`:



Public Member Functions

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

6.51.1 Detailed Description

```
template<int FilterOrder, class StateType = DEFAULT_STATE>
struct lir::ChebyshevII::HighPass< FilterOrder, StateType >
```

[ChebyshevII](#) highpass filter

Parameters

<i>FilterOrder</i>	The order of the filter.
<i>StateType</i>	The filter topology: DirectFormI , DirectFormII , ...

6.51.2 Member Function Documentation

6.51.2.1 setup()

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

Calculates the coefficients of the filter

Parameters

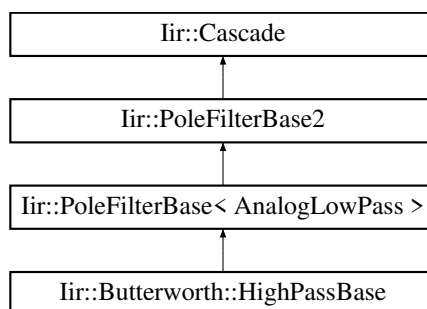
<i>sampleRate</i>	Sampling rate
<i>cutoffFrequency</i>	Cutoff frequency.
<i>stopBandDb</i>	Permitted ripples in dB in the stopband

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

- iir/ChebyshevII.h

6.52 Iir::Butterworth::HighPassBase Struct Reference

Inheritance diagram for Iir::Butterworth::HighPassBase:

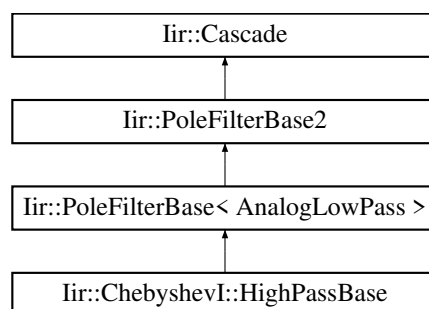


The documentation for this struct was generated from the following files:

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

6.53 Iir::ChebyshevI::HighPassBase Struct Reference

Inheritance diagram for Iir::ChebyshevI::HighPassBase:

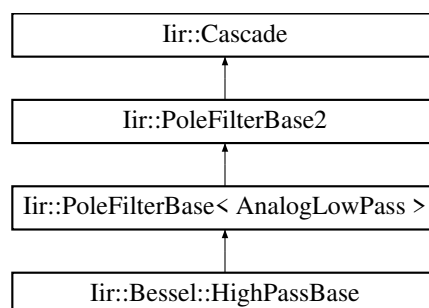


The documentation for this struct was generated from the following files:

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

6.54 Iir::Bessel::HighPassBase Struct Reference

Inheritance diagram for Iir::Bessel::HighPassBase:

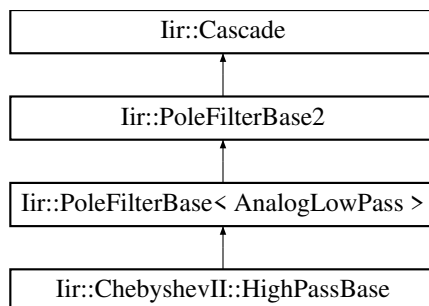


The documentation for this struct was generated from the following files:

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

6.55 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

6.56 Iir::HighPassTransform Class Reference

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

6.56.1 Detailed Description

low pass to high pass

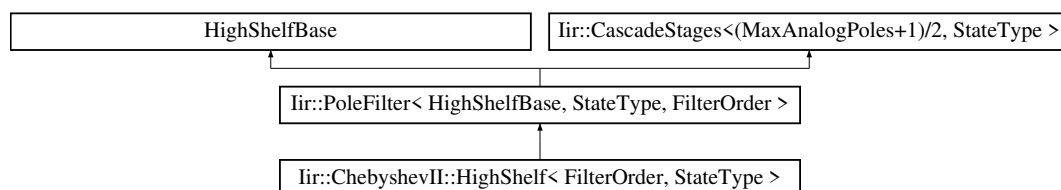
The documentation for this class was generated from the following files:

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

6.57 Iir::ChebyshevII::HighShelf< FilterOrder, StateType > Struct Template Reference

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

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



Public Member Functions

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

6.57.1 Detailed Description

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

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

Parameters

<i>FilterOrder</i>	The order of the filter.
<i>StateType</i>	The filter topology: DirectFormI , DirectFormII , ...

6.57.2 Member Function Documentation

6.57.2.1 setup()

```
template<int FilterOrder, class StateType = DEFAULT_STATE>
void Iir::ChebyshevII::HighShelf< FilterOrder, 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

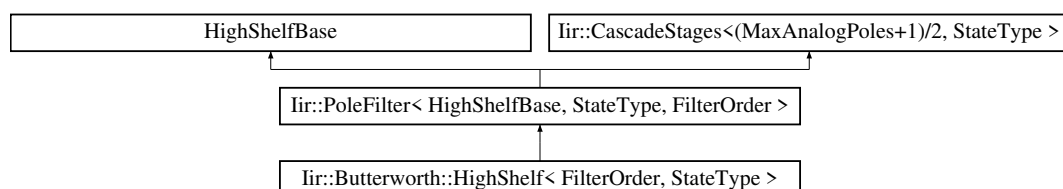
The documentation for this struct was generated from the following file:

- iir/ChebyshevII.h

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

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

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



Public Member Functions

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

6.58.1 Detailed Description

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

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

Parameters

<i>FilterOrder</i>	The order of the filter.
<i>StateType</i>	The filter topology: DirectFormI , DirectFormII , ...

6.58.2 Member Function Documentation

6.58.2.1 setup()

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

Calculates the coefficients

Parameters

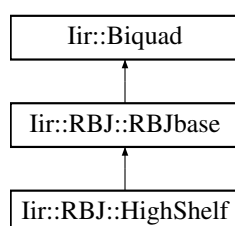
<i>sampleRate</i>	Sampling rate
<i>cutoffFrequency</i>	Cutoff
<i>gainDb</i>	Gain in dB of the filter in the passband

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

- [iir/Butterworth.h](#)

6.59 Iir::RBJ::HighShelf Struct Reference

Inheritance diagram for Iir::RBJ::HighShelf:



Additional Inherited Members

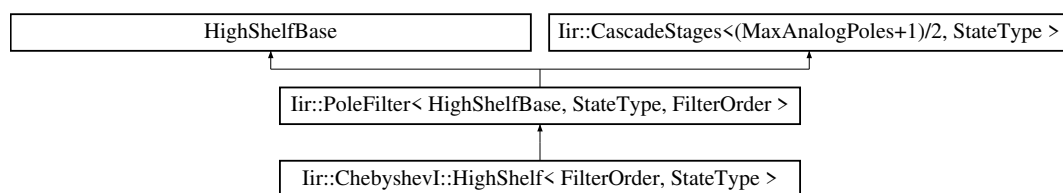
The documentation for this struct was generated from the following files:

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

6.60 Iir::ChebyshevI::HighShelf< FilterOrder, StateType > Struct Template Reference

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

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



Public Member Functions

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

6.60.1 Detailed Description

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

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

Parameters

<i>FilterOrder</i>	The order of the filter.
<i>StateType</i>	The filter topology: DirectFormI , DirectFormII , ...

6.60.2 Member Function Documentation

6.60.2.1 setup()

```
template<int FilterOrder, class StateType = DEFAULT_STATE>
void Iir::ChebyshevI::HighShelf< FilterOrder, StateType >::setup (
    double sampleRate,
```

```
double cutoffFrequency,
double gainDb,
double rippleDb ) [inline]
```

Calculates the coefficients of the filter

Parameters

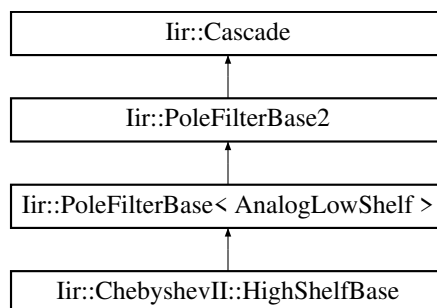
<i>sampleRate</i>	Sampling rate
<i>cutoffFrequency</i>	Cutoff frequency.
<i>gainDb</i>	Gain in the passband
<i>rippleDb</i>	Permitted ripples in dB in the passband

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

- iir/ChebyshevI.h

6.61 Iir::ChebyshevII::HighShelfBase Struct Reference

Inheritance diagram for Iir::ChebyshevII::HighShelfBase:

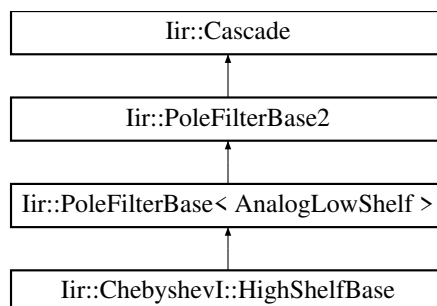


The documentation for this struct was generated from the following files:

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

6.62 Iir::ChebyshevI::HighShelfBase Struct Reference

Inheritance diagram for Iir::ChebyshevI::HighShelfBase:

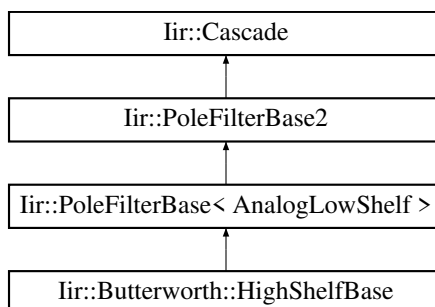


The documentation for this struct was generated from the following files:

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

6.63 Iir::Butterworth::HighShelfBase Struct Reference

Inheritance diagram for Iir::Butterworth::HighShelfBase:



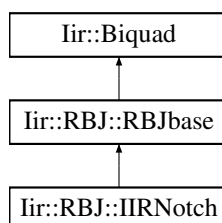
The documentation for this struct was generated from the following files:

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

6.64 Iir::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=10)

6.64.1 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 $r = \exp(-\pi \cdot (\text{centerFrequency}/\text{sampleRate})/q_factor)$ whereas the angles of the poles/zeros define the bandstop frequency. The higher Q the closer r moves towards the unit circle.

6.64.2 Member Function Documentation

6.64.2.1 setup()

```
void Iir::RBJ::IIRNotch::setup (
    double sampleRate,
    double centerFrequency,
    double q_factor = 10 )
```

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

6.65 Iir::Layout< MaxPoles > Class Template Reference

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

6.65.1 Detailed Description

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

Storage for [Layout](#)

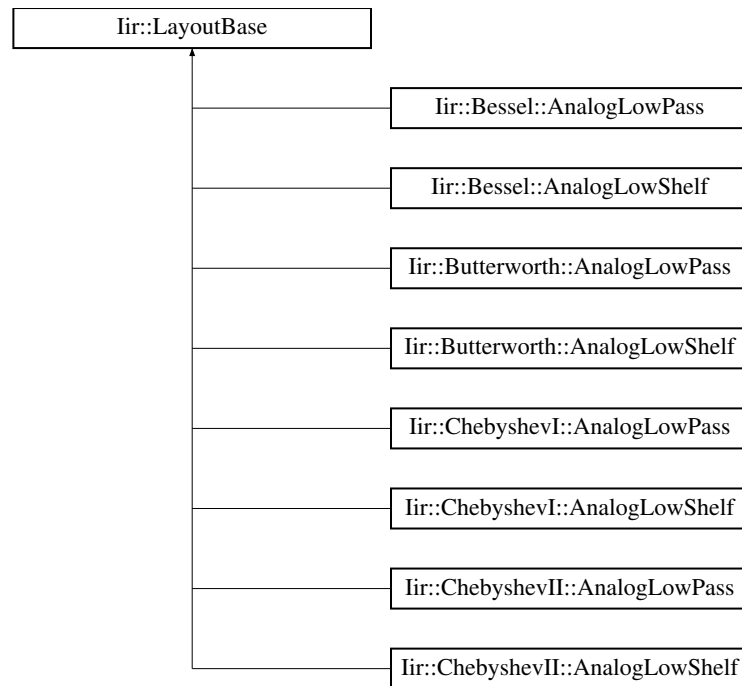
The documentation for this class was generated from the following file:

- iir/Layout.h

6.66 Iir::LayoutBase Class Reference

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

Inheritance diagram for Iir::LayoutBase:



6.66.1 Detailed Description

Base uses pointers to reduce template instantiations

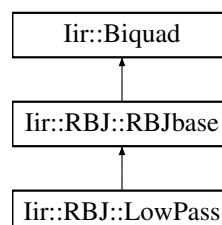
The documentation for this class was generated from the following file:

- iir/Layout.h

6.67 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)

6.67.1 Detailed Description

Lowpass.

6.67.2 Member Function Documentation

6.67.2.1 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.

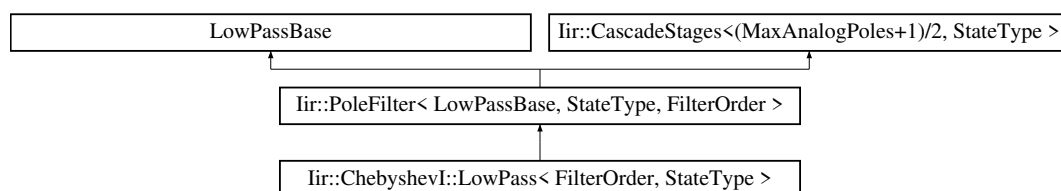
The documentation for this struct was generated from the following files:

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

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

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

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



Public Member Functions

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

6.68.1 Detailed Description

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

[ChebyshevI](#) lowpass filter

Parameters

<i>FilterOrder</i>	The order of the filter.
<i>StateType</i>	The filter topology: DirectFormI , DirectFormII , ...

6.68.2 Member Function Documentation

6.68.2.1 setup()

```
template<int FilterOrder, class StateType = DEFAULT_STATE>
void Iir::ChebyshevI::LowPass< FilterOrder, 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

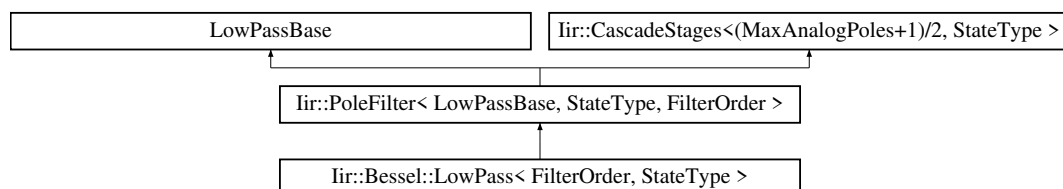
The documentation for this struct was generated from the following file:

- iir/ChebyshevI.h

6.69 Iir::Bessel::LowPass< FilterOrder, StateType > Struct Template Reference

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

Inheritance diagram for Iir::Bessel::LowPass< FilterOrder, StateType >:



Public Member Functions

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

6.69.1 Detailed Description

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

[Bessel](#) Lowpass

Parameters

<i>FilterOrder</i>	The order of the filter.
<i>StateType</i>	The filter topology: DirectFormI , DirectFormII , ...

6.69.2 Member Function Documentation

6.69.2.1 setup()

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

Calculate the coefficients

Parameters

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

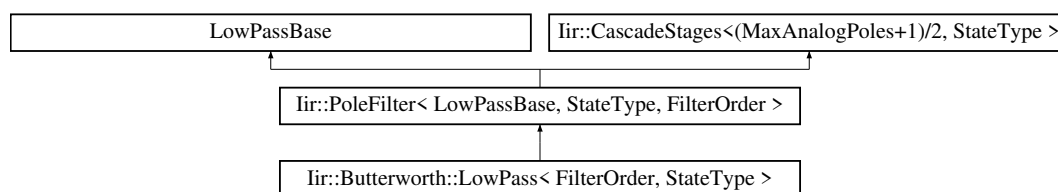
The documentation for this struct was generated from the following file:

- iir/Bessel.h

6.70 Iir::Butterworth::LowPass< FilterOrder, StateType > Struct Template Reference

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

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



Public Member Functions

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

6.70.1 Detailed Description

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

[Butterworth](#) Lowpass filter.

Parameters

<i>FilterOrder</i>	The order of the filter.
<i>StateType</i>	The filter topology: DirectFormI , DirectFormII , ...

6.70.2 Member Function Documentation

6.70.2.1 `setup()`

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

Calculates the coefficients

Parameters

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

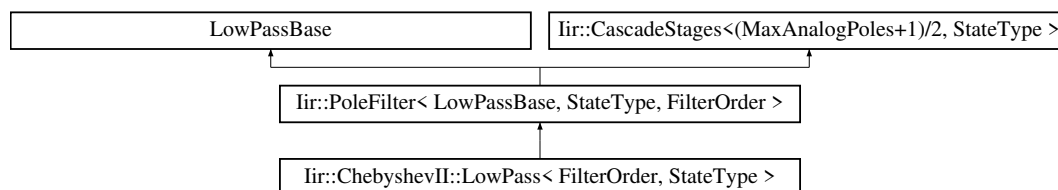
The documentation for this struct was generated from the following file:

- `iir/Butterworth.h`

6.71 `Iir::ChebyshevII::LowPass< FilterOrder, StateType >` Struct Template Reference

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

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



Public Member Functions

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

6.71.1 Detailed Description

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

[ChebyshevII](#) lowpass filter

Parameters

<i>FilterOrder</i>	The order of the filter.
<i>StateType</i>	The filter topology: DirectFormI , DirectFormII , ...

6.71.2 Member Function Documentation

6.71.2.1 setup()

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

Calculates the coefficients of the filter

Parameters

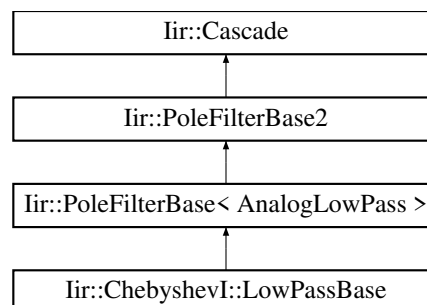
<i>sampleRate</i>	Sampling rate
<i>cutoffFrequency</i>	Cutoff frequency.
<i>stopBandDb</i>	Permitted ripples in dB in the stopband

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

- iir/ChebyshevII.h

6.72 Iir::ChebyshevI::LowPassBase Struct Reference

Inheritance diagram for Iir::ChebyshevI::LowPassBase:

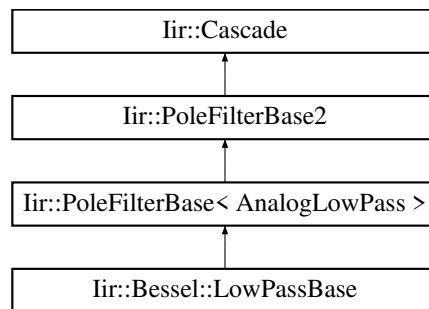


The documentation for this struct was generated from the following files:

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

6.73 Iir::Bessel::LowPassBase Struct Reference

Inheritance diagram for Iir::Bessel::LowPassBase:

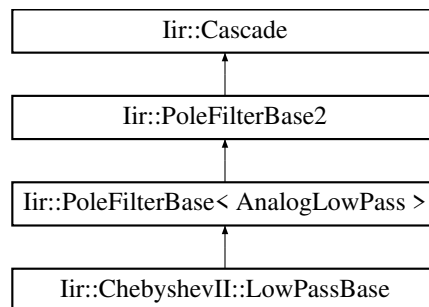


The documentation for this struct was generated from the following files:

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

6.74 Iir::ChebyshevII::LowPassBase Struct Reference

Inheritance diagram for Iir::ChebyshevII::LowPassBase:

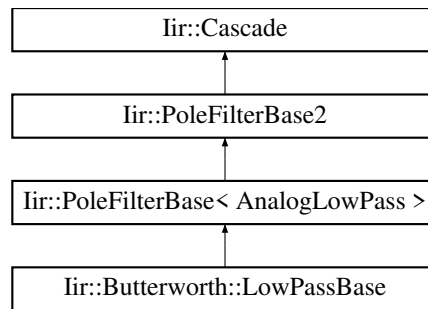


The documentation for this struct was generated from the following files:

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

6.75 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

6.76 Iir::LowPassTransform Class Reference

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

6.76.1 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

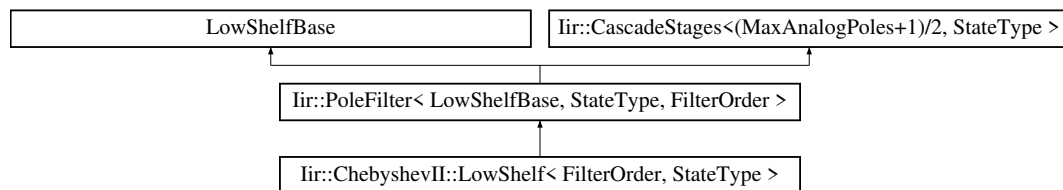
The documentation for this class was generated from the following files:

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

6.77 Iir::ChebyshevII::LowShelf< FilterOrder, StateType > Struct Template Reference

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

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



Public Member Functions

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

6.77.1 Detailed Description

```
template<int FilterOrder, class StateType = DEFAULT_STATE>
```

```
struct Iir::ChebyshevII::LowShelf< FilterOrder, StateType >
```

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

Parameters

<i>FilterOrder</i>	The order of the filter.
<i>StateType</i>	The filter topology: DirectFormI , DirectFormII , ...

6.77.2 Member Function Documentation

6.77.2.1 `setup()`

```
template<int FilterOrder, class StateType = DEFAULT_STATE>
void Iir::ChebyshevII::LowShelf< FilterOrder, 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

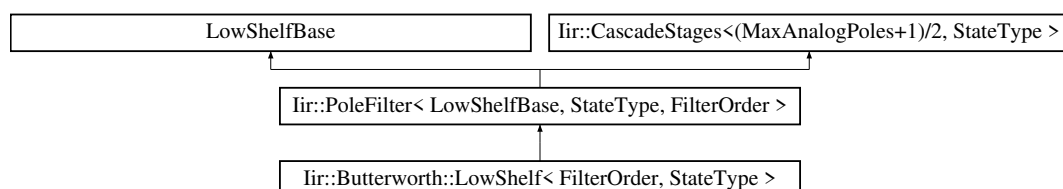
The documentation for this struct was generated from the following file:

- `iir/ChebyshevII.h`

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

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

Inheritance diagram for `Iir::Butterworth::LowShelf< FilterOrder, StateType >`:



Public Member Functions

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

6.78.1 Detailed Description

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

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

Parameters

<i>FilterOrder</i>	The order of the filter.
<i>StateType</i>	The filter topology: DirectFormI , DirectFormII , ...

6.78.2 Member Function Documentation

6.78.2.1 setup()

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

Calculates the coefficients

Parameters

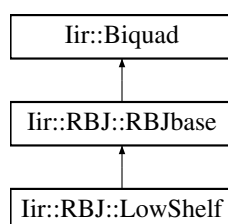
<i>sampleRate</i>	Sampling rate
<i>cutoffFrequency</i>	Cutoff
<i>gainDb</i>	Gain in dB of the filter in the passband

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

- [iir/Butterworth.h](#)

6.79 Iir::RBJ::LowShelf Struct Reference

Inheritance diagram for Iir::RBJ::LowShelf:



Additional Inherited Members

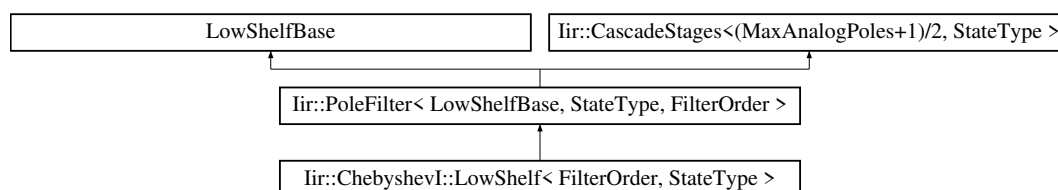
The documentation for this struct was generated from the following files:

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

6.80 iir::ChebyshevI::LowShelf< FilterOrder, StateType > Struct Template Reference

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

Inheritance diagram for iir::ChebyshevI::LowShelf< FilterOrder, StateType >:



Public Member Functions

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

6.80.1 Detailed Description

```
template<int FilterOrder, class StateType = DEFAULT_STATE>
struct iir::ChebyshevI::LowShelf< FilterOrder, StateType >
```

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

Parameters

<i>FilterOrder</i>	The order of the filter.
<i>StateType</i>	The filter topology: DirectFormI , DirectFormII , ...

6.80.2 Member Function Documentation

6.80.2.1 setup()

```
template<int FilterOrder, class StateType = DEFAULT_STATE>
void iir::ChebyshevI::LowShelf< FilterOrder, StateType >::setup (
    double sampleRate,
```

```
double cutoffFrequency,
double gainDb,
double rippleDb ) [inline]
```

Calculates the coefficients of the filter

Parameters

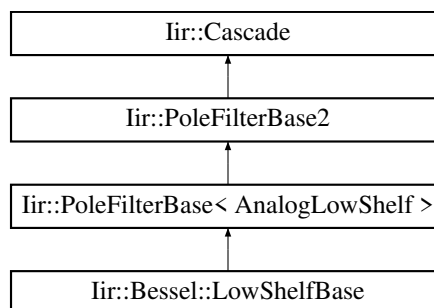
<i>sampleRate</i>	Sampling rate
<i>cutoffFrequency</i>	Cutoff frequency.
<i>gainDb</i>	Gain in the passband
<i>rippleDb</i>	Permitted ripples in dB in the passband

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

- iir/ChebyshevI.h

6.81 Iir::Bessel::LowShelfBase Struct Reference

Inheritance diagram for Iir::Bessel::LowShelfBase:

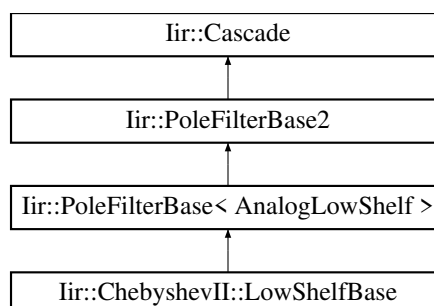


The documentation for this struct was generated from the following files:

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

6.82 Iir::ChebyshevII::LowShelfBase Struct Reference

Inheritance diagram for Iir::ChebyshevII::LowShelfBase:

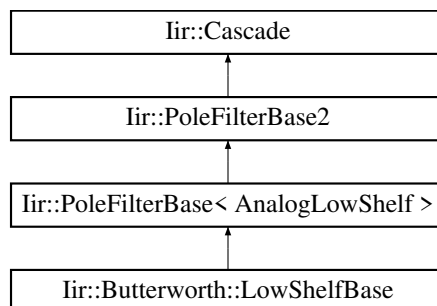


The documentation for this struct was generated from the following files:

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

6.83 Iir::Butterworth::LowShelfBase Struct Reference

Inheritance diagram for Iir::Butterworth::LowShelfBase:

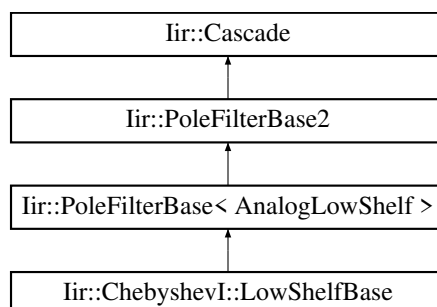


The documentation for this struct was generated from the following files:

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

6.84 Iir::ChebyshevI::LowShelfBase Struct Reference

Inheritance diagram for Iir::ChebyshevI::LowShelfBase:



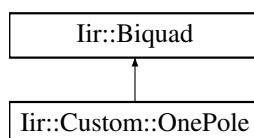
The documentation for this struct was generated from the following files:

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

6.85 Iir::Custom::OnePole Struct Reference

```
#include <Custom.h>
```

Inheritance diagram for Iir::Custom::OnePole:



Additional Inherited Members

6.85.1 Detailed Description

Setting up a filter with with one real pole, real zero and scale it by the scale factor

Parameters

<i>scale</i>	Scale the FIR coefficients by this factor
<i>pole</i>	Position of the pole on the real axis
<i>zero</i>	Position of the zero on the real axis

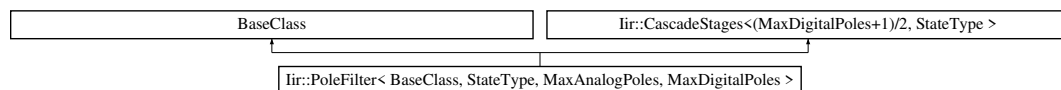
The documentation for this struct was generated from the following files:

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

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

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

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



Additional Inherited Members

6.86.1 Detailed Description

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

Storage for pole filters

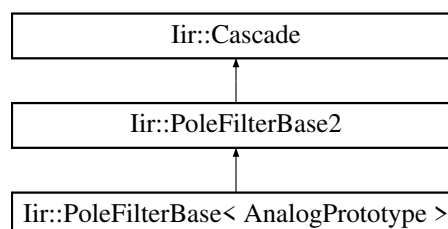
The documentation for this struct was generated from the following file:

- iir/PoleFilter.h

6.87 Iir::PoleFilterBase< AnalogPrototype > Class Template Reference

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

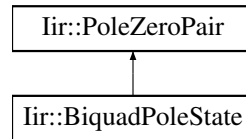
Inheritance diagram for Iir::PoleFilterBase< AnalogPrototype >:



6.89 Iir::PoleZeroPair Struct Reference

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

Inheritance diagram for Iir::PoleZeroPair:



6.89.1 Detailed Description

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

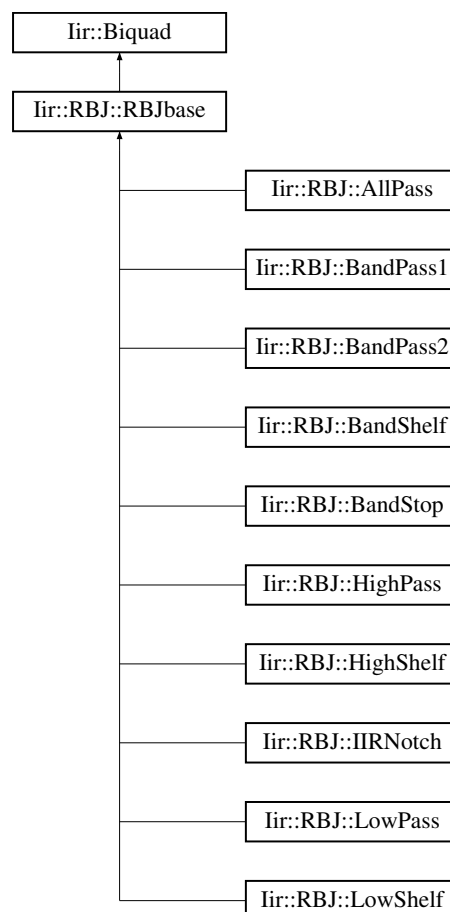
The documentation for this struct was generated from the following file:

- iir/Types.h

6.90 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

6.90.1 Detailed Description

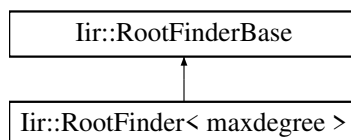
The base class of all [RBJ](#) filters

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

- `iir/RBJ.h`

6.91 Iir::RootFinder< maxdegree > Struct Template Reference

Inheritance diagram for `Iir::RootFinder< maxdegree >`:



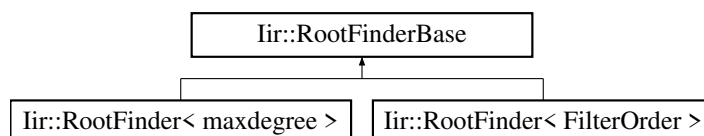
The documentation for this struct was generated from the following file:

- `iir/RootFinder.h`

6.92 Iir::RootFinderBase Class Reference

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

Inheritance diagram for `Iir::RootFinderBase`:



Classes

- struct [Array](#)

6.92.1 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

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

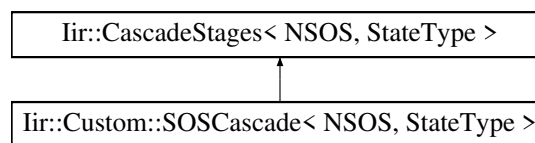
The documentation for this class was generated from the following file:

- iir/Utilities.h

6.94 Iir::Custom::SOSCascade< NSOS, StateType > Struct Template Reference

```
#include <Custom.h>
```

Inheritance diagram for Iir::Custom::SOSCascade< NSOS, StateType >:



Public Member Functions

- void [setup](#) (const double(&sosCoefficients)[NSOS][6])

6.94.1 Detailed Description

```
template<int NSOS, class StateType = DEFAULT_STATE>
struct Iir::Custom::SOSCascade< NSOS, StateType >
```

A custom cascade of 2nd order (SOS) filters.

Parameters

<i>NSOS</i>	The number of 2nd order filters.
<i>StateType</i>	The filter topology: DirectFormI , DirectFormII , ...

6.94.2 Member Function Documentation

6.94.2.1 setup()

```
template<int NSOS, class StateType = DEFAULT_STATE>
void Iir::Custom::SOSCascade< NSOS, StateType >::setup (
    const double(&) sosCoefficients[NSOS][6] ) [inline]
```

Python scipy.signal-friendly setting of coefficients. Sets the coefficients of the whole chain of biquads. The argument is a 2D array where the 1st dimension holds an array of 2nd order biquad coefficients. The sos coefficients are ordered "Python" style with first the FIR coefficients (B) and then the IIR coefficients (A). The const double array needs to have exactly the size [NSOS][6].

Parameters

<i>sosCoefficients</i>	2D array Python style sos[NSOS][6]. Indexing: 0-2: FIR, 3-5: IIR coefficients.
------------------------	--

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

- iir/Custom.h

6.95 Iir::TransposedDirectFormII Class Reference

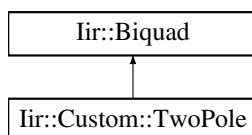
The documentation for this class was generated from the following file:

- iir/State.h

6.96 Iir::Custom::TwoPole Struct Reference

```
#include <Custom.h>
```

Inheritance diagram for Iir::Custom::TwoPole:



Additional Inherited Members

6.96.1 Detailed Description

Set a pole/zero pair in polar coordinates and scale the FIR filter coefficients

Parameters

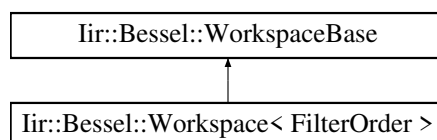
<i>poleRho</i>	Radius of the pole
<i>poleTheta</i>	Angle of the pole
<i>zeroRho</i>	Radius of the zero
<i>zeroTheta</i>	Angle of the zero

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

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

6.97 Iir::Bessel::Workspace< FilterOrder > Struct Template Reference

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

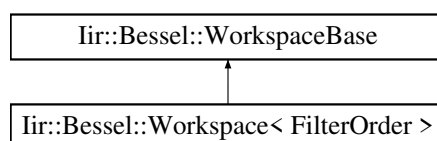


The documentation for this struct was generated from the following file:

- iir/Bessel.h

6.98 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, 15](#)
- applyScale
 - [lir::Biquad, 43](#)
- copy
 - [lir, 15](#)
- filter
 - [lir::Biquad, 43](#)
 - [lir::CascadeStages, 47](#)
- getA0
 - [lir::Biquad, 43](#)
- getA1
 - [lir::Biquad, 44](#)
- getA2
 - [lir::Biquad, 44](#)
- getB0
 - [lir::Biquad, 44](#)
- getB1
 - [lir::Biquad, 44](#)
- getB2
 - [lir::Biquad, 44](#)
- getPoleZeros
 - [lir::Biquad, 44](#)
- [lir, 13](#)
 - [add, 15](#)
 - [copy, 15](#)
 - [Kind, 15](#)
- [lir::BandPassTransform, 29](#)
- [lir::BandStopTransform, 42](#)
- [lir::Bessel, 16](#)
- [lir::Bessel::AnalogLowPass, 19](#)
- [lir::Bessel::AnalogLowShelf, 20](#)
- [lir::Bessel::BandPass](#)
 - [setup, 24](#)
- [lir::Bessel::BandPass< FilterOrder, StateType >, 24](#)
- [lir::Bessel::BandPassBase, 29](#)
- [lir::Bessel::BandStop](#)
 - [setup, 37](#)
- [lir::Bessel::BandStop< FilterOrder, StateType >, 37](#)
- [lir::Bessel::BandStopBase, 40](#)
- [lir::Bessel::HighPass](#)
 - [setup, 52](#)
- [lir::Bessel::HighPass< FilterOrder, StateType >, 52](#)
- [lir::Bessel::HighPassBase, 57](#)
- [lir::Bessel::LowPass](#)
 - [setup, 69](#)
- [lir::Bessel::LowPass< FilterOrder, StateType >, 67](#)
- [lir::Bessel::LowPassBase, 72](#)
- [lir::Bessel::LowShelfBase, 77](#)
- [lir::Bessel::Workspace< FilterOrder >, 85](#)
- [lir::Bessel::WorkspaceBase, 85](#)
- [lir::Biquad, 42](#)
- [applyScale, 43](#)
- [filter, 43](#)
- [getA0, 43](#)
- [getA1, 44](#)
- [getA2, 44](#)
- [getB0, 44](#)
- [getB1, 44](#)
- [getB2, 44](#)
- [getPoleZeros, 44](#)
- [response, 44](#)
- [setCoefficients, 45](#)
- [setIdentity, 45](#)
- [setOnePole, 45](#)
- [setPoleZeroPair, 45](#)
- [setTwoPole, 45](#)
- [lir::BiquadPoleState, 46](#)
- [lir::Butterworth, 16](#)
- [lir::Butterworth::AnalogLowPass, 19](#)
- [lir::Butterworth::AnalogLowShelf, 20](#)
- [lir::Butterworth::BandPass](#)
 - [setup, 23](#)
- [lir::Butterworth::BandPass< FilterOrder, StateType >, 22](#)
- [lir::Butterworth::BandPassBase, 28](#)
- [lir::Butterworth::BandShelf](#)
 - [setup, 33](#)
- [lir::Butterworth::BandShelf< FilterOrder, StateType >, 32](#)
- [lir::Butterworth::BandShelfBase, 34](#)
- [lir::Butterworth::BandStop](#)
 - [setup, 38](#)
- [lir::Butterworth::BandStop< FilterOrder, StateType >, 38](#)
- [lir::Butterworth::BandStopBase, 41](#)
- [lir::Butterworth::HighPass](#)
 - [setup, 54](#)
- [lir::Butterworth::HighPass< FilterOrder, StateType >, 54](#)
- [lir::Butterworth::HighPassBase, 56](#)
- [lir::Butterworth::HighShelf](#)
 - [setup, 60](#)
- [lir::Butterworth::HighShelf< FilterOrder, StateType >, 59](#)
- [lir::Butterworth::HighShelfBase, 63](#)
- [lir::Butterworth::LowPass](#)
 - [setup, 70](#)
- [lir::Butterworth::LowPass< FilterOrder, StateType >, 69](#)
- [lir::Butterworth::LowPassBase, 72](#)
- [lir::Butterworth::LowShelf](#)
 - [setup, 75](#)
- [lir::Butterworth::LowShelf< FilterOrder, StateType >, 74](#)
- [lir::Butterworth::LowShelfBase, 78](#)
- [lir::Cascade, 46](#)
- [lir::CascadeStages](#)
 - [filter, 47](#)

- reset, [49](#)
- setup, [49](#)
- lir::CascadeStages< MaxStages, StateType >, [47](#)
- lir::ChebyshevI::AnalogLowPass, [19](#)
- lir::ChebyshevI::AnalogLowShelf, [21](#)
- lir::ChebyshevI::BandPass
 - setup, [22](#)
- lir::ChebyshevI::BandPass< FilterOrder, StateType >, [21](#)
- lir::ChebyshevI::BandPassBase, [28](#)
- lir::ChebyshevI::BandShelf
 - setup, [32](#)
- lir::ChebyshevI::BandShelf< FilterOrder, StateType >, [31](#)
- lir::ChebyshevI::BandShelfBase, [33](#)
- lir::ChebyshevI::BandStop
 - setup, [36](#)
- lir::ChebyshevI::BandStop< FilterOrder, StateType >, [36](#)
- lir::ChebyshevI::BandStopBase, [40](#)
- lir::ChebyshevI::HighPass
 - setup, [53](#)
- lir::ChebyshevI::HighPass< FilterOrder, StateType >, [53](#)
- lir::ChebyshevI::HighPassBase, [57](#)
- lir::ChebyshevI::HighShelf
 - setup, [61](#)
- lir::ChebyshevI::HighShelf< FilterOrder, StateType >, [61](#)
- lir::ChebyshevI::HighShelfBase, [62](#)
- lir::ChebyshevI::LowPass
 - setup, [67](#)
- lir::ChebyshevI::LowPass< FilterOrder, StateType >, [66](#)
- lir::ChebyshevI::LowPassBase, [71](#)
- lir::ChebyshevI::LowShelf
 - setup, [76](#)
- lir::ChebyshevI::LowShelf< FilterOrder, StateType >, [76](#)
- lir::ChebyshevI::LowShelfBase, [78](#)
- lir::ChebyshevII::AnalogLowPass, [20](#)
- lir::ChebyshevII::AnalogLowShelf, [21](#)
- lir::ChebyshevII::BandPass
 - setup, [25](#)
- lir::ChebyshevII::BandPass< FilterOrder, StateType >, [25](#)
- lir::ChebyshevII::BandPassBase, [29](#)
- lir::ChebyshevII::BandShelf
 - setup, [30](#)
- lir::ChebyshevII::BandShelf< FilterOrder, StateType >, [30](#)
- lir::ChebyshevII::BandShelfBase, [34](#)
- lir::ChebyshevII::BandStop
 - setup, [39](#)
- lir::ChebyshevII::BandStop< FilterOrder, StateType >, [39](#)
- lir::ChebyshevII::BandStopBase, [41](#)
- lir::ChebyshevII::HighPass
 - setup, [56](#)
- lir::ChebyshevII::HighPass< FilterOrder, StateType >, [55](#)
- lir::ChebyshevII::HighPassBase, [57](#)
- lir::ChebyshevII::HighShelf
 - setup, [59](#)
- lir::ChebyshevII::HighShelf< FilterOrder, StateType >, [58](#)
- lir::ChebyshevII::HighShelfBase, [62](#)
- lir::ChebyshevII::LowPass
 - setup, [71](#)
- lir::ChebyshevII::LowPass< FilterOrder, StateType >, [70](#)
- lir::ChebyshevII::LowPassBase, [72](#)
- lir::ChebyshevII::LowShelf
 - setup, [74](#)
- lir::ChebyshevII::LowShelf< FilterOrder, StateType >, [73](#)
- lir::ChebyshevII::LowShelfBase, [77](#)
- lir::ChebyshevII, [17](#)
- lir::ChebyshevI, [17](#)
- lir::ComplexPair, [49](#)
- lir::Custom::OnePole, [78](#)
- lir::Custom::SOSCascade
 - setup, [84](#)
- lir::Custom::SOSCascade< NSOS, StateType >, [83](#)
- lir::Custom::TwoPole, [84](#)
- lir::DirectFormII, [50](#)
- lir::DirectFormI, [50](#)
- lir::EnvelopeFollower< Channels, Value >, [50](#)
- lir::HighPassTransform, [58](#)
- lir::Layout< MaxPoles >, [64](#)
- lir::LayoutBase, [64](#)
- lir::LowPassTransform, [73](#)
- lir::PoleFilter< BaseClass, StateType, MaxAnalogPoles, MaxDigitalPoles >, [79](#)
- lir::PoleFilterBase< AnalogPrototype >, [79](#)
- lir::PoleFilterBase2, [80](#)
- lir::PoleZeroPair, [81](#)
- lir::RBJ::AllPass, [18](#)
- lir::RBJ::BandPass1, [26](#)
 - setup, [26](#)
- lir::RBJ::BandPass2, [27](#)
 - setup, [27](#)
- lir::RBJ::BandShelf, [31](#)
- lir::RBJ::BandStop, [35](#)
 - setup, [35](#)
- lir::RBJ::HighPass, [51](#)
 - setup, [51](#)
- lir::RBJ::HighShelf, [60](#)
- lir::RBJ::IIRNotch, [63](#)
 - setup, [63](#)
- lir::RBJ::LowPass, [65](#)
 - setup, [66](#)
- lir::RBJ::LowShelf, [75](#)
- lir::RBJ::RBJbase, [81](#)
- lir::RBJ, [18](#)
- lir::RootFinder< maxdegree >, [82](#)
- lir::RootFinderBase, [82](#)

`lir::RootFinderBase::Array`, 21
`lir::SlopeDetector< Channels, Value >`, 83
`lir::TransposedDirectFormII`, 84

Kind

`lir`, 15

reset

`lir::CascadeStages`, 49

response

`lir::Biquad`, 44

setCoefficients

`lir::Biquad`, 45

setIdentity

`lir::Biquad`, 45

setOnePole

`lir::Biquad`, 45

setPoleZeroPair

`lir::Biquad`, 45

setTwoPole

`lir::Biquad`, 45

setup

`lir::Bessel::BandPass`, 24

`lir::Bessel::BandStop`, 37

`lir::Bessel::HighPass`, 52

`lir::Bessel::LowPass`, 69

`lir::Butterworth::BandPass`, 23

`lir::Butterworth::BandShelf`, 33

`lir::Butterworth::BandStop`, 38

`lir::Butterworth::HighPass`, 54

`lir::Butterworth::HighShelf`, 60

`lir::Butterworth::LowPass`, 70

`lir::Butterworth::LowShelf`, 75

`lir::CascadeStages`, 49

`lir::ChebyshevI::BandPass`, 22

`lir::ChebyshevI::BandShelf`, 32

`lir::ChebyshevI::BandStop`, 36

`lir::ChebyshevI::HighPass`, 53

`lir::ChebyshevI::HighShelf`, 61

`lir::ChebyshevI::LowPass`, 67

`lir::ChebyshevI::LowShelf`, 76

`lir::ChebyshevII::BandPass`, 25

`lir::ChebyshevII::BandShelf`, 30

`lir::ChebyshevII::BandStop`, 39

`lir::ChebyshevII::HighPass`, 56

`lir::ChebyshevII::HighShelf`, 59

`lir::ChebyshevII::LowPass`, 71

`lir::ChebyshevII::LowShelf`, 74

`lir::Custom::SOSCascade`, 84

`lir::RBJ::BandPass1`, 26

`lir::RBJ::BandPass2`, 27

`lir::RBJ::BandStop`, 35

`lir::RBJ::HighPass`, 51

`lir::RBJ::IIRNotch`, 63

`lir::RBJ::LowPass`, 66