

iir1

Generated by Doxygen 1.8.13

## Contents

<b>1</b>	<b>IIR1 -- Realtime C++ filter library</b>	<b>2</b>
<b>2</b>	<b>Namespace Index</b>	<b>4</b>
2.1	Namespace List . . . . .	4
<b>3</b>	<b>Hierarchical Index</b>	<b>5</b>
3.1	Class Hierarchy . . . . .	5
<b>4</b>	<b>Class Index</b>	<b>10</b>
4.1	Class List . . . . .	10
<b>5</b>	<b>Namespace Documentation</b>	<b>14</b>
5.1	lir::Bessel Namespace Reference . . . . .	14
5.1.1	Detailed Description . . . . .	14
5.2	lir::Butterworth Namespace Reference . . . . .	14
5.2.1	Detailed Description . . . . .	15
5.3	lir::ChebyshevI Namespace Reference . . . . .	15
5.3.1	Detailed Description . . . . .	15
5.4	lir::ChebyshevII Namespace Reference . . . . .	15
5.4.1	Detailed Description . . . . .	16
5.5	lir::Elliptic Namespace Reference . . . . .	16
5.5.1	Detailed Description . . . . .	16
5.6	lir::Legendre Namespace Reference . . . . .	16
5.6.1	Detailed Description . . . . .	16
5.7	lir::RBJ Namespace Reference . . . . .	17
5.7.1	Detailed Description . . . . .	17

<b>6</b>	<b>Class Documentation</b>	<b>17</b>
6.1	<a href="#">lir::RBJ::AllPass Struct Reference</a>	17
6.2	<a href="#">lir::Butterworth::AnalogLowPass Class Reference</a>	18
6.3	<a href="#">lir::ChebyshevI::AnalogLowPass Class Reference</a>	18
6.4	<a href="#">lir::Elliptic::AnalogLowPass Class Reference</a>	18
6.5	<a href="#">lir::Legendre::AnalogLowPass Class Reference</a>	19
6.6	<a href="#">lir::Bessel::AnalogLowPass Class Reference</a>	19
6.7	<a href="#">lir::ChebyshevII::AnalogLowPass Class Reference</a>	19
6.8	<a href="#">lir::Bessel::AnalogLowShelf Class Reference</a>	20
6.9	<a href="#">lir::Butterworth::AnalogLowShelf Class Reference</a>	20
6.10	<a href="#">lir::ChebyshevI::AnalogLowShelf Class Reference</a>	20
6.11	<a href="#">lir::ChebyshevII::AnalogLowShelf Class Reference</a>	21
6.12	<a href="#">lir::RootFinderBase::Array Struct Reference</a>	21
6.13	<a href="#">lir::ChebyshevII::BandPass&lt; MaxOrder, StateType &gt; Struct Template Reference</a>	21
6.13.1	Detailed Description	21
6.13.2	Member Function Documentation	22
6.14	<a href="#">lir::Elliptic::BandPass&lt; MaxOrder, StateType &gt; Struct Template Reference</a>	22
6.14.1	Member Function Documentation	22
6.15	<a href="#">lir::Legendre::BandPass&lt; MaxOrder, StateType &gt; Struct Template Reference</a>	23
6.16	<a href="#">lir::Bessel::BandPass&lt; MaxOrder, StateType &gt; Struct Template Reference</a>	23
6.16.1	Detailed Description	24
6.16.2	Member Function Documentation	24
6.17	<a href="#">lir::ChebyshevI::BandPass&lt; MaxOrder, StateType &gt; Struct Template Reference</a>	24
6.17.1	Detailed Description	25
6.17.2	Member Function Documentation	25
6.18	<a href="#">lir::Butterworth::BandPass&lt; MaxOrder, StateType &gt; Struct Template Reference</a>	25
6.18.1	Detailed Description	26
6.18.2	Member Function Documentation	26
6.19	<a href="#">lir::RBJ::BandPass1 Struct Reference</a>	26
6.19.1	Detailed Description	27

6.19.2	Member Function Documentation	27
6.20	<code>lir::RBJ::BandPass2</code> Struct Reference	27
6.20.1	Detailed Description	28
6.20.2	Member Function Documentation	28
6.21	<code>lir::Elliptic::BandPassBase</code> Struct Reference	28
6.22	<code>lir::Butterworth::BandPassBase</code> Struct Reference	29
6.23	<code>lir::ChebyshevI::BandPassBase</code> Struct Reference	29
6.24	<code>lir::Legendre::BandPassBase</code> Struct Reference	29
6.25	<code>lir::Bessel::BandPassBase</code> Struct Reference	30
6.26	<code>lir::ChebyshevII::BandPassBase</code> Struct Reference	30
6.27	<code>lir::BandPassTransform</code> Class Reference	31
6.28	<code>lir::Butterworth::BandShelf&lt; MaxOrder, StateType &gt;</code> Struct Template Reference	31
6.28.1	Detailed Description	31
6.28.2	Member Function Documentation	31
6.29	<code>lir::ChebyshevII::BandShelf&lt; MaxOrder, StateType &gt;</code> Struct Template Reference	32
6.29.1	Detailed Description	32
6.29.2	Member Function Documentation	32
6.30	<code>lir::ChebyshevI::BandShelf&lt; MaxOrder, StateType &gt;</code> Struct Template Reference	33
6.30.1	Detailed Description	33
6.30.2	Member Function Documentation	33
6.31	<code>lir::RBJ::BandShelf</code> Struct Reference	34
6.32	<code>lir::ChebyshevI::BandShelfBase</code> Struct Reference	34
6.33	<code>lir::Butterworth::BandShelfBase</code> Struct Reference	35
6.34	<code>lir::ChebyshevII::BandShelfBase</code> Struct Reference	35
6.35	<code>lir::ChebyshevII::BandStop&lt; MaxOrder, StateType &gt;</code> Struct Template Reference	35
6.35.1	Detailed Description	36
6.35.2	Member Function Documentation	36
6.36	<code>lir::Elliptic::BandStop&lt; MaxOrder, StateType &gt;</code> Struct Template Reference	36
6.36.1	Member Function Documentation	37
6.37	<code>lir::Legendre::BandStop&lt; MaxOrder, StateType &gt;</code> Struct Template Reference	37

6.38	<a href="#">lir::ChebyshevI::BandStop&lt; MaxOrder, StateType &gt; Struct Template Reference</a>	38
6.38.1	Detailed Description	38
6.38.2	Member Function Documentation	38
6.39	<a href="#">lir::Bessel::BandStop&lt; MaxOrder, StateType &gt; Struct Template Reference</a>	39
6.39.1	Detailed Description	39
6.39.2	Member Function Documentation	39
6.40	<a href="#">lir::RBJ::BandStop Struct Reference</a>	40
6.40.1	Detailed Description	40
6.40.2	Member Function Documentation	40
6.41	<a href="#">lir::Butterworth::BandStop&lt; MaxOrder, StateType &gt; Struct Template Reference</a>	41
6.41.1	Detailed Description	41
6.41.2	Member Function Documentation	41
6.42	<a href="#">lir::Bessel::BandStopBase Struct Reference</a>	42
6.43	<a href="#">lir::Elliptic::BandStopBase Struct Reference</a>	42
6.44	<a href="#">lir::ChebyshevI::BandStopBase Struct Reference</a>	42
6.45	<a href="#">lir::Legendre::BandStopBase Struct Reference</a>	43
6.46	<a href="#">lir::Butterworth::BandStopBase Struct Reference</a>	43
6.47	<a href="#">lir::ChebyshevII::BandStopBase Struct Reference</a>	44
6.48	<a href="#">lir::BandStopTransform Class Reference</a>	44
6.49	<a href="#">lir::Biquad Class Reference</a>	44
6.50	<a href="#">lir::BiquadBase Class Reference</a>	45
6.51	<a href="#">lir::BiquadPoleState Struct Reference</a>	45
6.52	<a href="#">lir::Cascade Class Reference</a>	46
6.53	<a href="#">lir::CascadeStages&lt; MaxStages, StateType &gt; Class Template Reference</a>	46
6.54	<a href="#">lir::ComplexPair Struct Reference</a>	47
6.55	<a href="#">lir::DirectFormI Class Reference</a>	47
6.56	<a href="#">lir::DirectFormII Class Reference</a>	47
6.57	<a href="#">lir::EnvelopeFollower&lt; Channels, Value &gt; Class Template Reference</a>	47
6.58	<a href="#">lir::ChebyshevII::HighPass&lt; MaxOrder, StateType &gt; Struct Template Reference</a>	47
6.58.1	Detailed Description	48

6.58.2	Member Function Documentation	48
6.59	<a href="#">lir::Butterworth::HighPass&lt; MaxOrder, StateType &gt; Struct Template Reference</a>	48
6.59.1	Detailed Description	49
6.59.2	Member Function Documentation	49
6.60	<a href="#">lir::Elliptic::HighPass&lt; MaxOrder, StateType &gt; Struct Template Reference</a>	49
6.60.1	Member Function Documentation	50
6.61	<a href="#">lir::Bessel::HighPass&lt; MaxOrder, StateType &gt; Struct Template Reference</a>	50
6.61.1	Detailed Description	50
6.61.2	Member Function Documentation	51
6.62	<a href="#">lir::Legendre::HighPass&lt; MaxOrder, StateType &gt; Struct Template Reference</a>	51
6.63	<a href="#">lir::ChebyshevI::HighPass&lt; MaxOrder, StateType &gt; Struct Template Reference</a>	51
6.63.1	Detailed Description	52
6.63.2	Member Function Documentation	52
6.64	<a href="#">lir::RBJ::HighPass Struct Reference</a>	52
6.64.1	Detailed Description	53
6.64.2	Member Function Documentation	53
6.65	<a href="#">lir::Elliptic::HighPassBase Struct Reference</a>	53
6.66	<a href="#">lir::ChebyshevI::HighPassBase Struct Reference</a>	54
6.67	<a href="#">lir::Butterworth::HighPassBase Struct Reference</a>	54
6.68	<a href="#">lir::Legendre::HighPassBase Struct Reference</a>	54
6.69	<a href="#">lir::Bessel::HighPassBase Struct Reference</a>	55
6.70	<a href="#">lir::ChebyshevII::HighPassBase Struct Reference</a>	55
6.71	<a href="#">lir::HighPassTransform Class Reference</a>	56
6.72	<a href="#">lir::Butterworth::HighShelf&lt; MaxOrder, StateType &gt; Struct Template Reference</a>	56
6.72.1	Detailed Description	56
6.72.2	Member Function Documentation	56
6.73	<a href="#">lir::ChebyshevII::HighShelf&lt; MaxOrder, StateType &gt; Struct Template Reference</a>	57
6.73.1	Detailed Description	57
6.73.2	Member Function Documentation	57
6.74	<a href="#">lir::ChebyshevI::HighShelf&lt; MaxOrder, StateType &gt; Struct Template Reference</a>	58

6.74.1 Detailed Description . . . . .	58
6.74.2 Member Function Documentation . . . . .	58
6.75 lir::RBJ::HighShelf Struct Reference . . . . .	59
6.76 lir::ChebyshevI::HighShelfBase Struct Reference . . . . .	59
6.77 lir::Butterworth::HighShelfBase Struct Reference . . . . .	60
6.78 lir::ChebyshevII::HighShelfBase Struct Reference . . . . .	60
6.79 lir::Layout< MaxPoles > Class Template Reference . . . . .	60
6.80 lir::LayoutBase Class Reference . . . . .	61
6.81 lir::ChebyshevII::LowPass< MaxOrder, StateType > Struct Template Reference . . . . .	61
6.81.1 Detailed Description . . . . .	62
6.81.2 Member Function Documentation . . . . .	62
6.82 lir::Elliptic::LowPass< MaxOrder, StateType > Struct Template Reference . . . . .	62
6.82.1 Member Function Documentation . . . . .	63
6.83 lir::Bessel::LowPass< MaxOrder, StateType > Struct Template Reference . . . . .	63
6.83.1 Detailed Description . . . . .	63
6.83.2 Member Function Documentation . . . . .	64
6.84 lir::Legendre::LowPass< MaxOrder, StateType > Struct Template Reference . . . . .	64
6.85 lir::ChebyshevI::LowPass< MaxOrder, StateType > Struct Template Reference . . . . .	64
6.85.1 Detailed Description . . . . .	65
6.85.2 Member Function Documentation . . . . .	65
6.86 lir::RBJ::LowPass Struct Reference . . . . .	65
6.86.1 Detailed Description . . . . .	66
6.86.2 Member Function Documentation . . . . .	66
6.87 lir::Butterworth::LowPass< MaxOrder, StateType > Struct Template Reference . . . . .	66
6.87.1 Detailed Description . . . . .	67
6.87.2 Member Function Documentation . . . . .	67
6.88 lir::ChebyshevI::LowPassBase Struct Reference . . . . .	67
6.89 lir::Elliptic::LowPassBase Struct Reference . . . . .	68
6.90 lir::Bessel::LowPassBase Struct Reference . . . . .	68
6.91 lir::Butterworth::LowPassBase Struct Reference . . . . .	68

6.92	<a href="#">lir::Legendre::LowPassBase Struct Reference</a>	69
6.93	<a href="#">lir::ChebyshevII::LowPassBase Struct Reference</a>	69
6.94	<a href="#">lir::LowPassTransform Class Reference</a>	70
6.95	<a href="#">lir::ChebyshevII::LowShelf&lt; MaxOrder, StateType &gt; Struct Template Reference</a>	70
6.95.1	Detailed Description	70
6.95.2	Member Function Documentation	70
6.96	<a href="#">lir::ChebyshevI::LowShelf&lt; MaxOrder, StateType &gt; Struct Template Reference</a>	71
6.96.1	Detailed Description	71
6.96.2	Member Function Documentation	71
6.97	<a href="#">lir::RBJ::LowShelf Struct Reference</a>	72
6.98	<a href="#">lir::Butterworth::LowShelf&lt; MaxOrder, StateType &gt; Struct Template Reference</a>	72
6.98.1	Detailed Description	73
6.98.2	Member Function Documentation	73
6.99	<a href="#">lir::ChebyshevII::LowShelfBase Struct Reference</a>	73
6.100	<a href="#">lir::Bessel::LowShelfBase Struct Reference</a>	74
6.101	<a href="#">lir::ChebyshevI::LowShelfBase Struct Reference</a>	74
6.102	<a href="#">lir::Butterworth::LowShelfBase Struct Reference</a>	74
6.103	<a href="#">lir::Custom::OnePole Struct Reference</a>	75
6.104	<a href="#">lir::PoleFilter&lt; BaseClass, StateType, MaxAnalogPoles, MaxDigitalPoles &gt; Struct Template Reference</a>	75
6.105	<a href="#">lir::PoleFilterBase&lt; AnalogPrototype &gt; Class Template Reference</a>	76
6.106	<a href="#">lir::PoleFilterBase2 Class Reference</a>	76
6.107	<a href="#">lir::PoleZeroPair Struct Reference</a>	77
6.108	<a href="#">lir::Legendre::PolynomialFinder&lt; maxN &gt; Class Template Reference</a>	77
6.109	<a href="#">lir::Legendre::PolynomialFinderBase Class Reference</a>	77
6.110	<a href="#">lir::RBJ::RBJbase Struct Reference</a>	78
6.110.1	Detailed Description	78
6.111	<a href="#">lir::RootFinder&lt; maxdegree &gt; Struct Template Reference</a>	79
6.112	<a href="#">lir::RootFinderBase Class Reference</a>	79
6.113	<a href="#">lir::SlopeDetector&lt; Channels, Value &gt; Class Template Reference</a>	79
6.114	<a href="#">lir::Elliptic::Solver Class Reference</a>	79
6.115	<a href="#">lir::Cascade::Stage Struct Reference</a>	80
6.116	<a href="#">lir::BiquadBase::State&lt; StateType &gt; Struct Template Reference</a>	80
6.117	<a href="#">lir::Cascade::Storage Struct Reference</a>	80
6.118	<a href="#">lir::TransposedDirectFormI Class Reference</a>	80
6.119	<a href="#">lir::TransposedDirectFormII Class Reference</a>	81
6.120	<a href="#">lir::Custom::TwoPole Struct Reference</a>	81
6.121	<a href="#">lir::Bessel::Workspace&lt; MaxOrder &gt; Struct Template Reference</a>	81
6.122	<a href="#">lir::Legendre::Workspace&lt; MaxOrder &gt; Struct Template Reference</a>	81
6.123	<a href="#">lir::Bessel::WorkspaceBase Struct Reference</a>	82
6.124	<a href="#">lir::Legendre::WorkspaceBase Struct Reference</a>	82



## 1 IIR1 -- Realtime C++ filter library

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

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

The filter uses templates so that its memory requirements are completely specified at compile time. This guarantees maximum performance and won't require malloc/new 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 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

```
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

```
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

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

#### 4. RBJ (2nd order 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. Elliptic

```
Iir::Elliptic::LowPass<order> f;
const float pass_ripple_db = 5; // dB
const float rolloff = 0.1;
f.setup (samplingrate,
        cutoff_frequency,
        passband_ripple_db,
        rolloff);
```

### Realtime filtering sample by sample

A sample `x` is processed by the filter with the `filter` command and then saved in `y`. The type can be either float or double:

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

This is executed at the sampling rate in an endless loop or event handler.

### Packages for Ubuntu (xenial / bionic):

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

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

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

### Compilation from source

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

#### Linux / Mac

##### Run

```
cmake .
```

which generates the Makefile. Then run:

```
make  
sudo make install
```

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

Both gcc and clang have been tested.

#### Windows

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

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

## Unit tests

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

## Documentation

### Overview

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

### Learn from the demos

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

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

### Detailed documentation

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

Run `doxygen` to generate the HTML documentation.

## 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. The filter design routines are identical to `DSPFilters`.

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:

<a href="#">lir::Bessel</a>	14
<a href="#">lir::Butterworth</a>	14
<a href="#">lir::ChebyshevI</a>	15

<a href="#">lir::ChebyshevI</a>	15
<a href="#">lir::Elliptic</a>	16
<a href="#">lir::Legendre</a>	16
<a href="#">lir::RBJ</a>	17

## 3 Hierarchical Index

### 3.1 Class Hierarchy

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

<b>lir::RootFinderBase::Array</b>	21
BandPassBase	
<b>lir::PoleFilter&lt; BandPassBase, StateType, MaxOrder, MaxOrder *2 &gt;</b>	75
<b>lir::Bessel::BandPass&lt; MaxOrder, StateType &gt;</b>	23
<b>lir::Butterworth::BandPass&lt; MaxOrder, StateType &gt;</b>	25
<b>lir::ChebyshevI::BandPass&lt; MaxOrder, StateType &gt;</b>	24
<b>lir::ChebyshevII::BandPass&lt; MaxOrder, StateType &gt;</b>	21
<b>lir::Elliptic::BandPass&lt; MaxOrder, StateType &gt;</b>	22
<b>lir::Legendre::BandPass&lt; MaxOrder, StateType &gt;</b>	23
<b>lir::BandPassTransform</b>	31
BandShelfBase	
<b>lir::PoleFilter&lt; BandShelfBase, StateType, MaxOrder, MaxOrder *2 &gt;</b>	75
<b>lir::Butterworth::BandShelf&lt; MaxOrder, StateType &gt;</b>	31
<b>lir::ChebyshevI::BandShelf&lt; MaxOrder, StateType &gt;</b>	33
<b>lir::ChebyshevII::BandShelf&lt; MaxOrder, StateType &gt;</b>	32
BandStopBase	
<b>lir::PoleFilter&lt; BandStopBase, StateType, MaxOrder, MaxOrder *2 &gt;</b>	75
<b>lir::Bessel::BandStop&lt; MaxOrder, StateType &gt;</b>	39
<b>lir::Butterworth::BandStop&lt; MaxOrder, StateType &gt;</b>	41
<b>lir::ChebyshevI::BandStop&lt; MaxOrder, StateType &gt;</b>	38
<b>lir::ChebyshevII::BandStop&lt; MaxOrder, StateType &gt;</b>	35
<b>lir::Elliptic::BandStop&lt; MaxOrder, StateType &gt;</b>	36
<b>lir::Legendre::BandStop&lt; MaxOrder, StateType &gt;</b>	37
<b>lir::BandStopTransform</b>	44

<b>lir::BiquadBase</b>	<b>45</b>
<b>lir::Biquad</b>	<b>44</b>
<b>lir::Cascade::Stage</b>	<b>80</b>
<b>lir::Custom::OnePole</b>	<b>75</b>
<b>lir::Custom::TwoPole</b>	<b>81</b>
<b>lir::RBJ::RBJbase</b>	<b>78</b>
<b>lir::RBJ::AllPass</b>	<b>17</b>
<b>lir::RBJ::BandPass1</b>	<b>26</b>
<b>lir::RBJ::BandPass2</b>	<b>27</b>
<b>lir::RBJ::BandShelf</b>	<b>34</b>
<b>lir::RBJ::BandStop</b>	<b>40</b>
<b>lir::RBJ::HighPass</b>	<b>52</b>
<b>lir::RBJ::HighShelf</b>	<b>59</b>
<b>lir::RBJ::LowPass</b>	<b>65</b>
<b>lir::RBJ::LowShelf</b>	<b>72</b>
<b>lir::Cascade</b>	<b>46</b>
<b>lir::PoleFilterBase2</b>	<b>76</b>
<b>lir::PoleFilterBase&lt; AnalogPrototype &gt;</b>	<b>76</b>
<b>lir::PoleFilterBase&lt; AnalogLowPass &gt;</b>	<b>76</b>
<b>lir::Bessel::BandPassBase</b>	<b>30</b>
<b>lir::Bessel::BandStopBase</b>	<b>42</b>
<b>lir::Bessel::HighPassBase</b>	<b>55</b>
<b>lir::Bessel::LowPassBase</b>	<b>68</b>
<b>lir::Butterworth::BandPassBase</b>	<b>29</b>
<b>lir::Butterworth::BandStopBase</b>	<b>43</b>
<b>lir::Butterworth::HighPassBase</b>	<b>54</b>
<b>lir::Butterworth::LowPassBase</b>	<b>68</b>
<b>lir::ChebyshevI::BandPassBase</b>	<b>29</b>
<b>lir::ChebyshevI::BandStopBase</b>	<b>42</b>
<b>lir::ChebyshevI::HighPassBase</b>	<b>54</b>
<b>lir::ChebyshevI::LowPassBase</b>	<b>67</b>
<b>lir::ChebyshevII::BandPassBase</b>	<b>30</b>

lir::ChebyshevII::BandStopBase	44
lir::ChebyshevII::HighPassBase	55
lir::ChebyshevII::LowPassBase	69
lir::Elliptic::BandPassBase	28
lir::Elliptic::BandStopBase	42
lir::Elliptic::HighPassBase	53
lir::Elliptic::LowPassBase	68
lir::Legendre::BandPassBase	29
lir::Legendre::BandStopBase	43
lir::Legendre::HighPassBase	54
lir::Legendre::LowPassBase	69
lir::PoleFilterBase< AnalogLowShelf >	76
lir::Bessel::LowShelfBase	74
lir::Butterworth::BandShelfBase	35
lir::Butterworth::HighShelfBase	60
lir::Butterworth::LowShelfBase	74
lir::ChebyshevI::BandShelfBase	34
lir::ChebyshevI::HighShelfBase	59
lir::ChebyshevI::LowShelfBase	74
lir::ChebyshevII::BandShelfBase	35
lir::ChebyshevII::HighShelfBase	60
lir::ChebyshevII::LowShelfBase	73
lir::CascadeStages< MaxStages, StateType >	46
lir::CascadeStages<(MaxAnalogPoles+1)/2, StateType >	46
lir::PoleFilter< HighPassBase, StateType, MaxOrder >	75
lir::Bessel::HighPass< MaxOrder, StateType >	50
lir::Butterworth::HighPass< MaxOrder, StateType >	48
lir::ChebyshevI::HighPass< MaxOrder, StateType >	51
lir::ChebyshevII::HighPass< MaxOrder, StateType >	47
lir::Elliptic::HighPass< MaxOrder, StateType >	49
lir::Legendre::HighPass< MaxOrder, StateType >	51
lir::PoleFilter< HighShelfBase, StateType, MaxOrder >	75

lir::Butterworth::HighShelf< MaxOrder, StateType >	56
lir::ChebyshevI::HighShelf< MaxOrder, StateType >	58
lir::ChebyshevII::HighShelf< MaxOrder, StateType >	57
lir::PoleFilter< LowPassBase, StateType, MaxOrder >	75
lir::Bessel::LowPass< MaxOrder, StateType >	63
lir::Butterworth::LowPass< MaxOrder, StateType >	66
lir::ChebyshevI::LowPass< MaxOrder, StateType >	64
lir::ChebyshevII::LowPass< MaxOrder, StateType >	61
lir::Elliptic::LowPass< MaxOrder, StateType >	62
lir::Legendre::LowPass< MaxOrder, StateType >	64
lir::PoleFilter< LowShelfBase, StateType, MaxOrder >	75
lir::Butterworth::LowShelf< MaxOrder, StateType >	72
lir::ChebyshevI::LowShelf< MaxOrder, StateType >	71
lir::ChebyshevII::LowShelf< MaxOrder, StateType >	70
lir::CascadeStages<(MaxDigitalPoles+1)/2, StateType >	46
lir::PoleFilter< BaseClass, StateType, MaxAnalogPoles, MaxDigitalPoles >	75
lir::PoleFilter< BandPassBase, StateType, MaxOrder, MaxOrder *2 >	75
lir::PoleFilter< BandShelfBase, StateType, MaxOrder, MaxOrder *2 >	75
lir::PoleFilter< BandStopBase, StateType, MaxOrder, MaxOrder *2 >	75
complex_pair_t	
lir::ComplexPair	47
lir::DirectFormI	47
lir::DirectFormII	47
lir::EnvelopeFollower< Channels, Value >	47
HighPassBase	
lir::PoleFilter< HighPassBase, StateType, MaxOrder >	75
lir::HighPassTransform	56
HighShelfBase	
lir::PoleFilter< HighShelfBase, StateType, MaxOrder >	75
lir::Layout< MaxPoles >	60
lir::Layout< MaxAnalogPoles >	60
lir::Layout< MaxDigitalPoles >	60
lir::LayoutBase	61

<b>lir::Bessel::AnalogLowPass</b>	<b>19</b>
<b>lir::Bessel::AnalogLowShelf</b>	<b>20</b>
<b>lir::Butterworth::AnalogLowPass</b>	<b>18</b>
<b>lir::Butterworth::AnalogLowShelf</b>	<b>20</b>
<b>lir::ChebyshevI::AnalogLowPass</b>	<b>18</b>
<b>lir::ChebyshevI::AnalogLowShelf</b>	<b>20</b>
<b>lir::ChebyshevII::AnalogLowPass</b>	<b>19</b>
<b>lir::ChebyshevII::AnalogLowShelf</b>	<b>21</b>
<b>lir::Elliptic::AnalogLowPass</b>	<b>18</b>
<b>lir::Legendre::AnalogLowPass</b>	<b>19</b>
LowPassBase	
<b>lir::PoleFilter&lt; LowPassBase, StateType, MaxOrder &gt;</b>	<b>75</b>
<b>lir::LowPassTransform</b>	<b>70</b>
LowShelfBase	
<b>lir::PoleFilter&lt; LowShelfBase, StateType, MaxOrder &gt;</b>	<b>75</b>
<b>lir::PoleZeroPair</b>	<b>77</b>
<b>lir::BiquadPoleState</b>	<b>45</b>
<b>lir::Legendre::PolynomialFinderBase</b>	<b>77</b>
<b>lir::Legendre::PolynomialFinder&lt; maxN &gt;</b>	<b>77</b>
<b>lir::Legendre::PolynomialFinder&lt; MaxOrder &gt;</b>	<b>77</b>
<b>lir::RootFinderBase</b>	<b>79</b>
<b>lir::RootFinder&lt; maxdegree &gt;</b>	<b>79</b>
<b>lir::RootFinder&lt; MaxOrder *2 &gt;</b>	<b>79</b>
<b>lir::RootFinder&lt; MaxOrder &gt;</b>	<b>79</b>
<b>lir::SlopeDetector&lt; Channels, Value &gt;</b>	<b>79</b>
<b>lir::Elliptic::Solver</b>	<b>79</b>
StateType	
<b>lir::BiquadBase::State&lt; StateType &gt;</b>	<b>80</b>
<b>lir::Cascade::Storage</b>	<b>80</b>
<b>lir::TransposedDirectFormI</b>	<b>80</b>
<b>lir::TransposedDirectFormII</b>	<b>81</b>
<b>lir::Bessel::WorkspaceBase</b>	<b>82</b>
<b>lir::Bessel::Workspace&lt; MaxOrder &gt;</b>	<b>81</b>



<b>lir::Legendre::WorkspaceBase</b>	<b>82</b>
<b>lir::Legendre::Workspace&lt; MaxOrder &gt;</b>	<b>81</b>
BaseClass	
<b>lir::PoleFilter&lt; BaseClass, StateType, MaxAnalogPoles, MaxDigitalPoles &gt;</b>	<b>75</b>

## 4 Class Index

### 4.1 Class List

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

<b>lir::RBJ::AllPass</b>	<b>17</b>
<b>lir::Butterworth::AnalogLowPass</b>	<b>18</b>
<b>lir::ChebyshevI::AnalogLowPass</b>	<b>18</b>
<b>lir::Elliptic::AnalogLowPass</b>	<b>18</b>
<b>lir::Legendre::AnalogLowPass</b>	<b>19</b>
<b>lir::Bessel::AnalogLowPass</b>	<b>19</b>
<b>lir::ChebyshevII::AnalogLowPass</b>	<b>19</b>
<b>lir::Bessel::AnalogLowShelf</b>	<b>20</b>
<b>lir::Butterworth::AnalogLowShelf</b>	<b>20</b>
<b>lir::ChebyshevI::AnalogLowShelf</b>	<b>20</b>
<b>lir::ChebyshevII::AnalogLowShelf</b>	<b>21</b>
<b>lir::RootFinderBase::Array</b>	<b>21</b>
<b>lir::ChebyshevII::BandPass&lt; MaxOrder, StateType &gt;</b>	<b>21</b>
<b>lir::Elliptic::BandPass&lt; MaxOrder, StateType &gt;</b>	<b>22</b>
<b>lir::Legendre::BandPass&lt; MaxOrder, StateType &gt;</b>	<b>23</b>
<b>lir::Bessel::BandPass&lt; MaxOrder, StateType &gt;</b>	<b>23</b>
<b>lir::ChebyshevI::BandPass&lt; MaxOrder, StateType &gt;</b>	<b>24</b>
<b>lir::Butterworth::BandPass&lt; MaxOrder, StateType &gt;</b>	<b>25</b>
<b>lir::RBJ::BandPass1</b>	<b>26</b>
<b>lir::RBJ::BandPass2</b>	<b>27</b>
<b>lir::Elliptic::BandPassBase</b>	<b>28</b>
<b>lir::Butterworth::BandPassBase</b>	<b>29</b>
<b>lir::ChebyshevI::BandPassBase</b>	<b>29</b>

<a href="#">lir::Legendre::BandPassBase</a>	29
<a href="#">lir::Bessel::BandPassBase</a>	30
<a href="#">lir::ChebyshevII::BandPassBase</a>	30
<a href="#">lir::BandPassTransform</a>	31
<a href="#">lir::Butterworth::BandShelf&lt; MaxOrder, StateType &gt;</a>	31
<a href="#">lir::ChebyshevII::BandShelf&lt; MaxOrder, StateType &gt;</a>	32
<a href="#">lir::ChebyshevI::BandShelf&lt; MaxOrder, StateType &gt;</a>	33
<a href="#">lir::RBJ::BandShelf</a>	34
<a href="#">lir::ChebyshevI::BandShelfBase</a>	34
<a href="#">lir::Butterworth::BandShelfBase</a>	35
<a href="#">lir::ChebyshevII::BandShelfBase</a>	35
<a href="#">lir::ChebyshevII::BandStop&lt; MaxOrder, StateType &gt;</a>	35
<a href="#">lir::Elliptic::BandStop&lt; MaxOrder, StateType &gt;</a>	36
<a href="#">lir::Legendre::BandStop&lt; MaxOrder, StateType &gt;</a>	37
<a href="#">lir::ChebyshevI::BandStop&lt; MaxOrder, StateType &gt;</a>	38
<a href="#">lir::Bessel::BandStop&lt; MaxOrder, StateType &gt;</a>	39
<a href="#">lir::RBJ::BandStop</a>	40
<a href="#">lir::Butterworth::BandStop&lt; MaxOrder, StateType &gt;</a>	41
<a href="#">lir::Bessel::BandStopBase</a>	42
<a href="#">lir::Elliptic::BandStopBase</a>	42
<a href="#">lir::ChebyshevI::BandStopBase</a>	42
<a href="#">lir::Legendre::BandStopBase</a>	43
<a href="#">lir::Butterworth::BandStopBase</a>	43
<a href="#">lir::ChebyshevII::BandStopBase</a>	44
<a href="#">lir::BandStopTransform</a>	44
<a href="#">lir::Biquad</a>	44
<a href="#">lir::BiquadBase</a>	45
<a href="#">lir::BiquadPoleState</a>	45
<a href="#">lir::Cascade</a>	46
<a href="#">lir::CascadeStages&lt; MaxStages, StateType &gt;</a>	46
<a href="#">lir::ComplexPair</a>	47
<a href="#">lir::DirectFormI</a>	47

<a href="#">lir::DirectFormII</a>	47
<a href="#">lir::EnvelopeFollower&lt; Channels, Value &gt;</a>	47
<a href="#">lir::ChebyshevII::HighPass&lt; MaxOrder, StateType &gt;</a>	47
<a href="#">lir::Butterworth::HighPass&lt; MaxOrder, StateType &gt;</a>	48
<a href="#">lir::Elliptic::HighPass&lt; MaxOrder, StateType &gt;</a>	49
<a href="#">lir::Bessel::HighPass&lt; MaxOrder, StateType &gt;</a>	50
<a href="#">lir::Legendre::HighPass&lt; MaxOrder, StateType &gt;</a>	51
<a href="#">lir::ChebyshevI::HighPass&lt; MaxOrder, StateType &gt;</a>	51
<a href="#">lir::RBJ::HighPass</a>	52
<a href="#">lir::Elliptic::HighPassBase</a>	53
<a href="#">lir::ChebyshevI::HighPassBase</a>	54
<a href="#">lir::Butterworth::HighPassBase</a>	54
<a href="#">lir::Legendre::HighPassBase</a>	54
<a href="#">lir::Bessel::HighPassBase</a>	55
<a href="#">lir::ChebyshevII::HighPassBase</a>	55
<a href="#">lir::HighPassTransform</a>	56
<a href="#">lir::Butterworth::HighShelf&lt; MaxOrder, StateType &gt;</a>	56
<a href="#">lir::ChebyshevII::HighShelf&lt; MaxOrder, StateType &gt;</a>	57
<a href="#">lir::ChebyshevI::HighShelf&lt; MaxOrder, StateType &gt;</a>	58
<a href="#">lir::RBJ::HighShelf</a>	59
<a href="#">lir::ChebyshevI::HighShelfBase</a>	59
<a href="#">lir::Butterworth::HighShelfBase</a>	60
<a href="#">lir::ChebyshevII::HighShelfBase</a>	60
<a href="#">lir::Layout&lt; MaxPoles &gt;</a>	60
<a href="#">lir::LayoutBase</a>	61
<a href="#">lir::ChebyshevII::LowPass&lt; MaxOrder, StateType &gt;</a>	61
<a href="#">lir::Elliptic::LowPass&lt; MaxOrder, StateType &gt;</a>	62
<a href="#">lir::Bessel::LowPass&lt; MaxOrder, StateType &gt;</a>	63
<a href="#">lir::Legendre::LowPass&lt; MaxOrder, StateType &gt;</a>	64
<a href="#">lir::ChebyshevI::LowPass&lt; MaxOrder, StateType &gt;</a>	64
<a href="#">lir::RBJ::LowPass</a>	65
<a href="#">lir::Butterworth::LowPass&lt; MaxOrder, StateType &gt;</a>	66

<a href="#">lir::ChebyshevI::LowPassBase</a>	67
<a href="#">lir::Elliptic::LowPassBase</a>	68
<a href="#">lir::Bessel::LowPassBase</a>	68
<a href="#">lir::Butterworth::LowPassBase</a>	68
<a href="#">lir::Legendre::LowPassBase</a>	69
<a href="#">lir::ChebyshevII::LowPassBase</a>	69
<a href="#">lir::LowPassTransform</a>	70
<a href="#">lir::ChebyshevII::LowShelf&lt; MaxOrder, StateType &gt;</a>	70
<a href="#">lir::ChebyshevI::LowShelf&lt; MaxOrder, StateType &gt;</a>	71
<a href="#">lir::RBJ::LowShelf</a>	72
<a href="#">lir::Butterworth::LowShelf&lt; MaxOrder, StateType &gt;</a>	72
<a href="#">lir::ChebyshevII::LowShelfBase</a>	73
<a href="#">lir::Bessel::LowShelfBase</a>	74
<a href="#">lir::ChebyshevI::LowShelfBase</a>	74
<a href="#">lir::Butterworth::LowShelfBase</a>	74
<a href="#">lir::Custom::OnePole</a>	75
<a href="#">lir::PoleFilter&lt; BaseClass, StateType, MaxAnalogPoles, MaxDigitalPoles &gt;</a>	75
<a href="#">lir::PoleFilterBase&lt; AnalogPrototype &gt;</a>	76
<a href="#">lir::PoleFilterBase2</a>	76
<a href="#">lir::PoleZeroPair</a>	77
<a href="#">lir::Legendre::PolynomialFinder&lt; maxN &gt;</a>	77
<a href="#">lir::Legendre::PolynomialFinderBase</a>	77
<a href="#">lir::RBJ::RBJbase</a>	
The base class of all RBJ filters	78
<a href="#">lir::RootFinder&lt; maxdegree &gt;</a>	79
<a href="#">lir::RootFinderBase</a>	79
<a href="#">lir::SlopeDetector&lt; Channels, Value &gt;</a>	79
<a href="#">lir::Elliptic::Solver</a>	79
<a href="#">lir::Cascade::Stage</a>	80
<a href="#">lir::BiquadBase::State&lt; StateType &gt;</a>	80
<a href="#">lir::Cascade::Storage</a>	80
<a href="#">lir::TransposedDirectFormI</a>	80

<a href="#">lir::TransposedDirectFormII</a>	81
<a href="#">lir::Custom::TwoPole</a>	81
<a href="#">lir::Bessel::Workspace&lt; MaxOrder &gt;</a>	81
<a href="#">lir::Legendre::Workspace&lt; MaxOrder &gt;</a>	81
<a href="#">lir::Bessel::WorkspaceBase</a>	82
<a href="#">lir::Legendre::WorkspaceBase</a>	82

## 5 Namespace Documentation

### 5.1 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](#)

#### 5.1.1 Detailed Description

Filters with [Bessel](#) response characteristics

### 5.2 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](#)

#### 5.2.1 Detailed Description

Filters with [Butterworth](#) response characteristics

### 5.3 `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](#)

#### 5.3.1 Detailed Description

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

### 5.4 `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](#)

### 5.4.1 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.

## 5.5 `lir::Elliptic` Namespace Reference

### Classes

- class [AnalogLowPass](#)
- struct [BandPass](#)
- struct [BandPassBase](#)
- struct [BandStop](#)
- struct [BandStopBase](#)
- struct [HighPass](#)
- struct [HighPassBase](#)
- struct [LowPass](#)
- struct [LowPassBase](#)
- class [Solver](#)

### 5.5.1 Detailed Description

Filters with [Elliptic](#) response characteristics

## 5.6 `lir::Legendre` Namespace Reference

### Classes

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

### 5.6.1 Detailed Description

Filters with [Legendre](#) / "Optimum-L" response characteristics

## 5.7 Iir::RBJ Namespace Reference

### Classes

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

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

#### 5.7.1 Detailed Description

Filter realizations based on Robert Bristol-Johnson formulae:

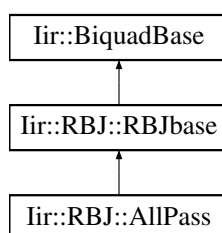
<http://www.musicdsp.org/files/Audio-EQ-Cookbook.txt>

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

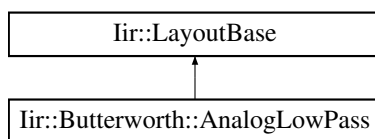
The documentation for this struct was generated from the following files:

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



## 6.2 Iir::Butterworth::AnalogLowPass Class Reference

Inheritance diagram for Iir::Butterworth::AnalogLowPass:

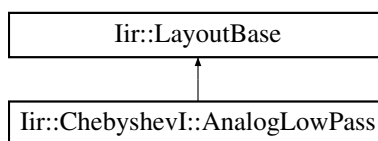


The documentation for this class was generated from the following files:

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

## 6.3 Iir::ChebyshevI::AnalogLowPass Class Reference

Inheritance diagram for Iir::ChebyshevI::AnalogLowPass:

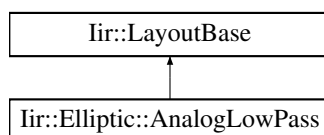


The documentation for this class was generated from the following files:

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

## 6.4 Iir::Elliptic::AnalogLowPass Class Reference

Inheritance diagram for Iir::Elliptic::AnalogLowPass:

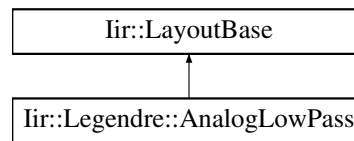


The documentation for this class was generated from the following files:

- iir/Elliptic.h
- iir/Elliptic.cpp

## 6.5 Iir::Legendre::AnalogLowPass Class Reference

Inheritance diagram for Iir::Legendre::AnalogLowPass:

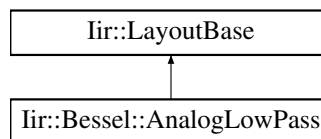


The documentation for this class was generated from the following files:

- iir/Legendre.h
- iir/Legendre.cpp

## 6.6 Iir::Bessel::AnalogLowPass Class Reference

Inheritance diagram for Iir::Bessel::AnalogLowPass:

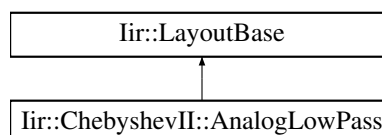


The documentation for this class was generated from the following files:

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

## 6.7 Iir::ChebyshevII::AnalogLowPass Class Reference

Inheritance diagram for Iir::ChebyshevII::AnalogLowPass:

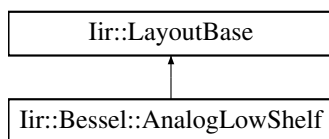


The documentation for this class was generated from the following files:

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

## 6.8 Iir::Bessel::AnalogLowShelf Class Reference

Inheritance diagram for Iir::Bessel::AnalogLowShelf:

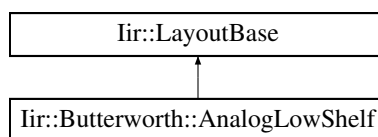


The documentation for this class was generated from the following files:

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

## 6.9 Iir::Butterworth::AnalogLowShelf Class Reference

Inheritance diagram for Iir::Butterworth::AnalogLowShelf:

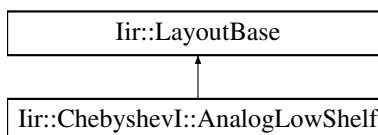


The documentation for this class was generated from the following files:

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

## 6.10 Iir::ChebyshevI::AnalogLowShelf Class Reference

Inheritance diagram for Iir::ChebyshevI::AnalogLowShelf:

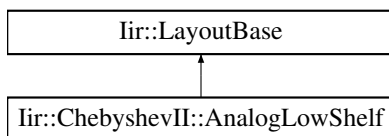


The documentation for this class was generated from the following files:

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

## 6.11 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

## 6.12 Iir::RootFinderBase::Array Struct Reference

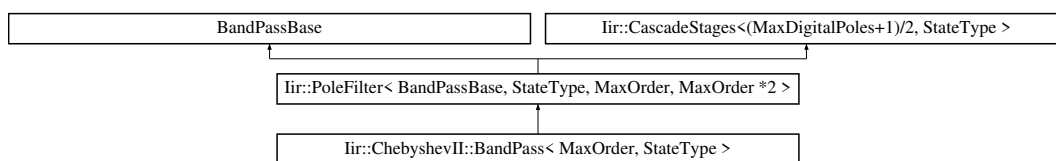
The documentation for this struct was generated from the following file:

- iir/RootFinder.h

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

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

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



### Public Member Functions

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

### 6.13.1 Detailed Description

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

[ChebyshevII](#) bandpass filter

### 6.13.2 Member Function Documentation

#### 6.13.2.1 setup()

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

Calculates the coefficients of the filter

#### Parameters

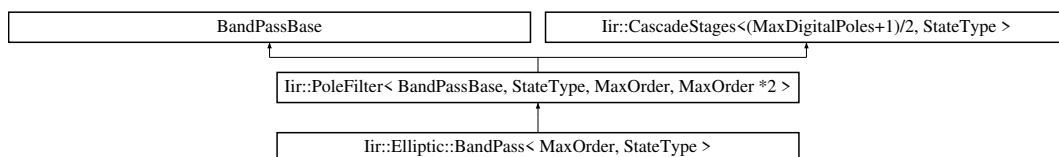
<i>sampleRate</i>	Sampling rate
<i>centerFrequency</i>	Center frequency of the bandpass
<i>widthFrequency</i>	Width of the bandpass
<i>stopBandDb</i>	Permitted ripples in dB in the stopband

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

- iir/ChebyshevII.h

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

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



#### Public Member Functions

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

#### 6.14.1 Member Function Documentation

## 6.14.1.1 setup()

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

Calculates the coefficients of the filter

## Parameters

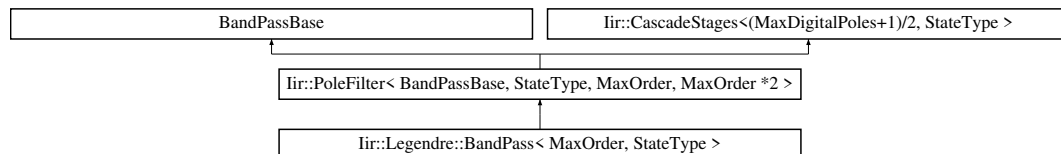
<i>sampleRate</i>	Sampling rate
<i>centerFrequency</i>	Centre frequency of the bandpass
<i>widthFrequency</i>	Frequency width of the bandpass
<i>rippleDb</i>	Permitted ripples in dB in the passband
<i>rolloff</i>	Rolloff from the pass- to stopband

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

- iir/Elliptic.h

## 6.15 Iir::Legendre::BandPass&lt; MaxOrder, StateType &gt; Struct Template Reference

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



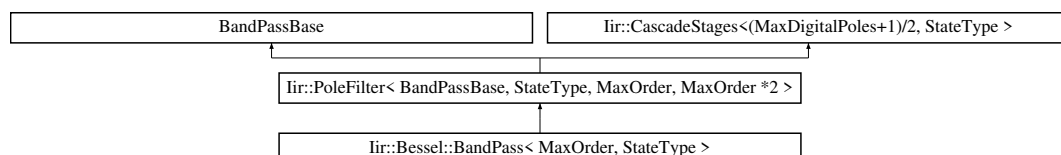
The documentation for this struct was generated from the following file:

- iir/Legendre.h

## 6.16 Iir::Bessel::BandPass&lt; MaxOrder, StateType &gt; Struct Template Reference

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

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



## Public Member Functions

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

### 6.16.1 Detailed Description

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

[Bessel](#) bandpass.

### 6.16.2 Member Function Documentation

#### 6.16.2.1 setup()

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

Calculate the coefficients

#### Parameters

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

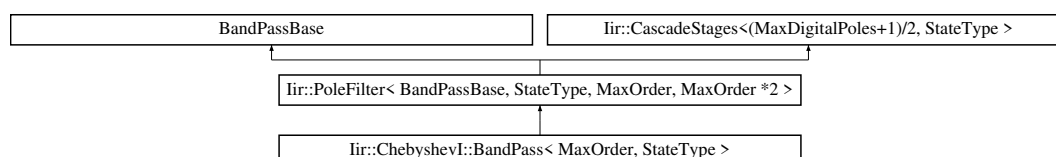
The documentation for this struct was generated from the following file:

- [iir/Bessel.h](#)

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

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

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



## Public Member Functions

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

## 6.17.1 Detailed Description

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

[ChebyshevI](#) bandpass filter

## 6.17.2 Member Function Documentation

## 6.17.2.1 setup()

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

Calculates the coefficients of the filter

## Parameters

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

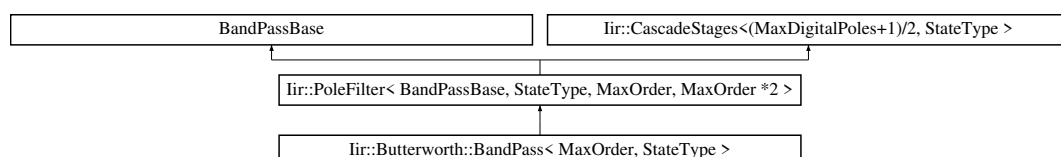
The documentation for this struct was generated from the following file:

- [iir/ChebyshevI.h](#)

## 6.18 Iir::Butterworth::BandPass&lt; MaxOrder, StateType &gt; Struct Template Reference

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

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





## Public Member Functions

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

### 6.18.1 Detailed Description

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

[Butterworth](#) Bandpass filter.

### 6.18.2 Member Function Documentation

#### 6.18.2.1 setup()

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

Calculates the coefficients

#### Parameters

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

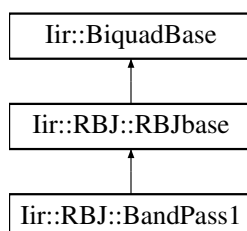
The documentation for this struct was generated from the following file:

- iir/Butterworth.h

## 6.19 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)

## 6.19.1 Detailed Description

Bandpass with constant skirt gain

## 6.19.2 Member Function Documentation

## 6.19.2.1 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

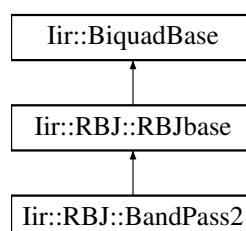
The documentation for this struct was generated from the following files:

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

## 6.20 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)

### 6.20.1 Detailed Description

Bandpass with constant 0 dB peak gain

### 6.20.2 Member Function Documentation

#### 6.20.2.1 setup()

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

Calculates the coefficients

#### Parameters

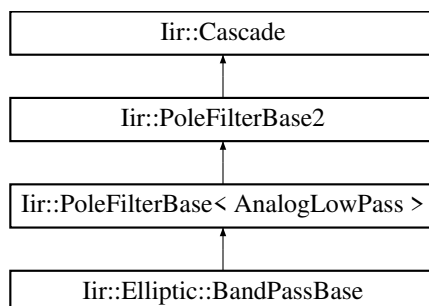
<i>sampleRate</i>	Sampling rate
<i>centerFrequency</i>	Center frequency of the bandpass
<i>bandWidth</i>	Bandwidth of the bandpass

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

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

## 6.21 Iir::Elliptic::BandPassBase Struct Reference

Inheritance diagram for Iir::Elliptic::BandPassBase:

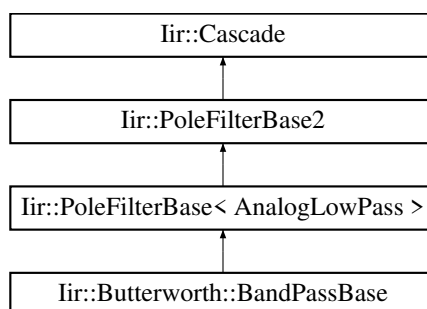


The documentation for this struct was generated from the following files:

- iir/Elliptic.h
- iir/Elliptic.cpp

## 6.22 Iir::Butterworth::BandPassBase Struct Reference

Inheritance diagram for Iir::Butterworth::BandPassBase:

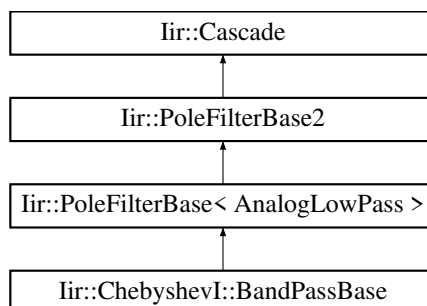


The documentation for this struct was generated from the following files:

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

## 6.23 Iir::ChebyshevI::BandPassBase Struct Reference

Inheritance diagram for Iir::ChebyshevI::BandPassBase:

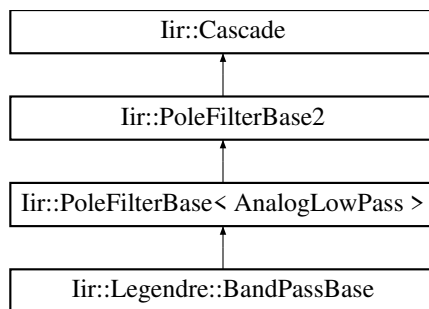


The documentation for this struct was generated from the following files:

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

## 6.24 Iir::Legendre::BandPassBase Struct Reference

Inheritance diagram for Iir::Legendre::BandPassBase:

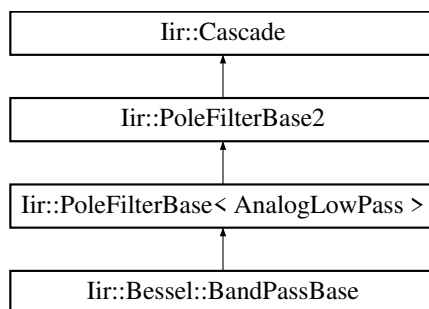


The documentation for this struct was generated from the following files:

- `iir/Legendre.h`
- `iir/Legendre.cpp`

## 6.25 Iir::Bessel::BandPassBase Struct Reference

Inheritance diagram for `Iir::Bessel::BandPassBase`:

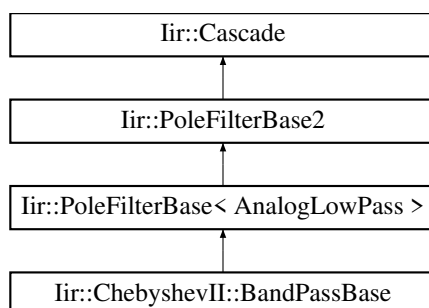


The documentation for this struct was generated from the following files:

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

## 6.26 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`

## 6.27 Iir::BandPassTransform Class Reference

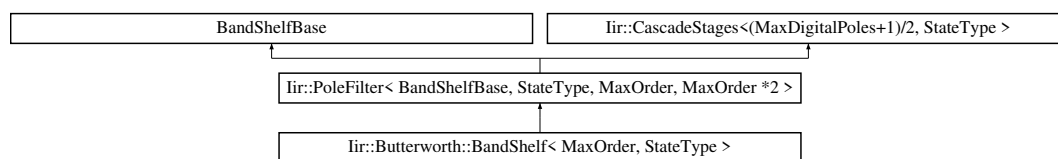
The documentation for this class was generated from the following files:

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

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

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

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



### Public Member Functions

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

#### 6.28.1 Detailed Description

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

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

#### 6.28.2 Member Function Documentation

##### 6.28.2.1 setup()

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

Calculates the coefficients

## Parameters

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

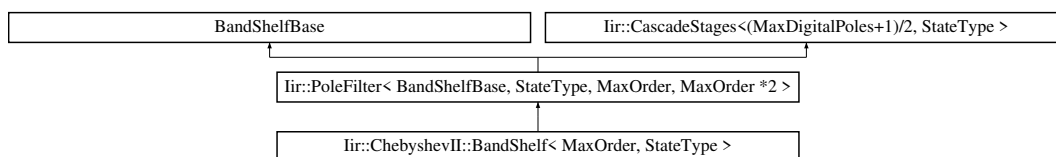
The documentation for this struct was generated from the following file:

- iir/Butterworth.h

## 6.29 `lir::ChebyshevII::BandShelf< MaxOrder, StateType >` Struct Template Reference

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

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



## Public Member Functions

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

### 6.29.1 Detailed Description

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

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

### 6.29.2 Member Function Documentation

#### 6.29.2.1 `setup()`

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

Calculates the coefficients of the filter

## Parameters

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

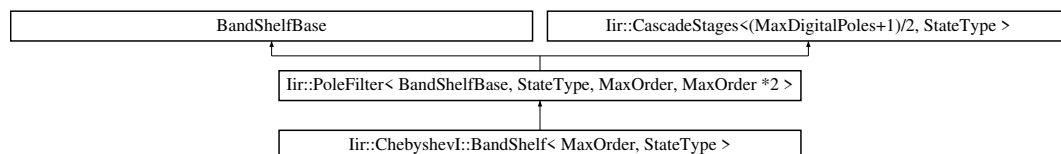
The documentation for this struct was generated from the following file:

- iir/ChebyshevI.h

## 6.30 Iir::ChebyshevI::BandShelf&lt; MaxOrder, StateType &gt; Struct Template Reference

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

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



## Public Member Functions

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

## 6.30.1 Detailed Description

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

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

## 6.30.2 Member Function Documentation

## 6.30.2.1 setup()

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

Calculates the coefficients of the filter



## Parameters

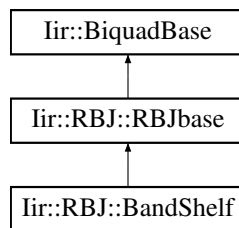
<i>sampleRate</i>	Sampling rate
<i>centerFrequency</i>	Center frequency of the passband
<i>widthFrequency</i>	Width of the passband.
<i>gainDb</i>	Gain in the passband. The stopband has 0 dB.
<i>rippleDb</i>	Permitted ripples in dB in the passband.

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

- iir/ChebyshevI.h

### 6.31 Iir::RBJ::BandShelf Struct Reference

Inheritance diagram for Iir::RBJ::BandShelf:



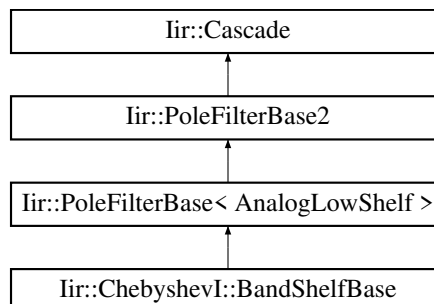
## Additional Inherited Members

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

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

### 6.32 Iir::ChebyshevI::BandShelfBase Struct Reference

Inheritance diagram for Iir::ChebyshevI::BandShelfBase:

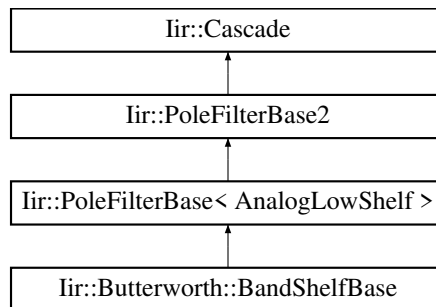


The documentation for this struct was generated from the following files:

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

### 6.33 Iir::Butterworth::BandShelfBase Struct Reference

Inheritance diagram for Iir::Butterworth::BandShelfBase:

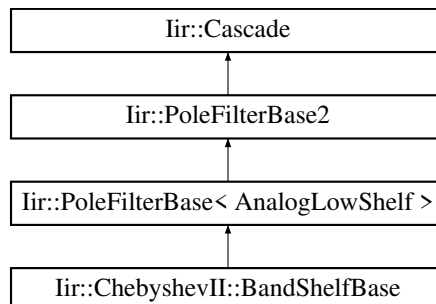


The documentation for this struct was generated from the following files:

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

### 6.34 Iir::ChebyshevII::BandShelfBase Struct Reference

Inheritance diagram for Iir::ChebyshevII::BandShelfBase:



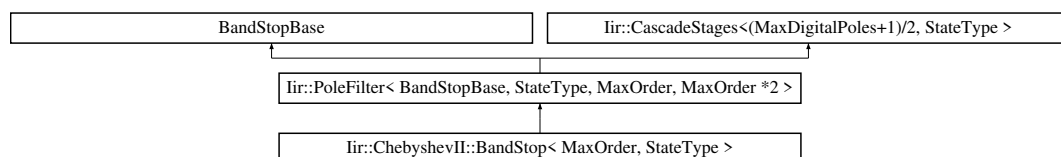
The documentation for this struct was generated from the following files:

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

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

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

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



## Public Member Functions

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

### 6.35.1 Detailed Description

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

[ChebyshevII](#) bandstop filter.

### 6.35.2 Member Function Documentation

#### 6.35.2.1 setup()

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

Calculates the coefficients of the filter

#### Parameters

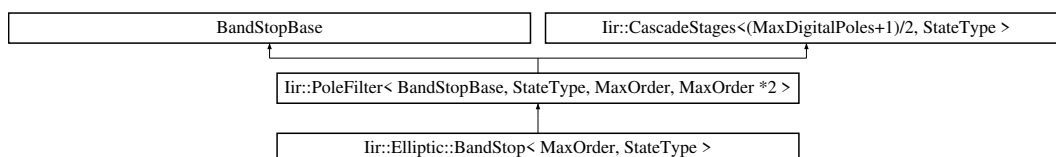
<i>sampleRate</i>	Sampling rate
<i>centerFrequency</i>	Center frequency of the bandstop
<i>widthFrequency</i>	Width of the bandstop
<i>stopBandDb</i>	Permitted ripples in dB in the stopband

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

- [iir/ChebyshevII.h](#)

## 6.36 Iir::Elliptic::BandStop< MaxOrder, StateType > Struct Template Reference

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



## Public Member Functions

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

## 6.36.1 Member Function Documentation

## 6.36.1.1 setup()

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

Calculates the coefficients of the filter

## Parameters

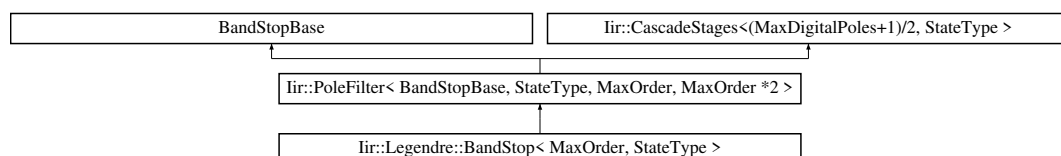
<i>sampleRate</i>	Sampling rate
<i>centerFrequency</i>	Centre frequency of the bandstop
<i>widthFrequency</i>	Frequency width of the bandstop
<i>rippleDb</i>	Permitted ripples in dB in the passband
<i>rolloff</i>	Rolloff from the pass- to stopband

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

- iir/Elliptic.h

## 6.37 Iir::Legendre::BandStop&lt; MaxOrder, StateType &gt; Struct Template Reference

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



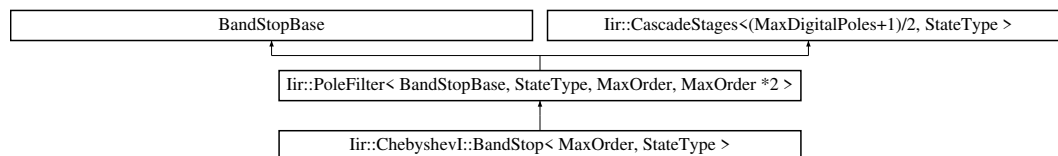
The documentation for this struct was generated from the following file:

- iir/Legendre.h

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

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

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



#### Public Member Functions

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

#### 6.38.1 Detailed Description

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

[ChebyshevI](#) bandstop filter

#### 6.38.2 Member Function Documentation

##### 6.38.2.1 setup()

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

Calculates the coefficients of the filter

#### Parameters

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

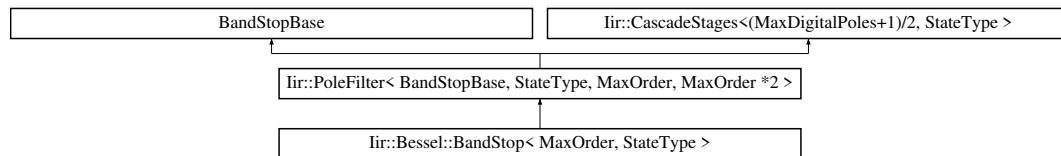
The documentation for this struct was generated from the following file:

- iir/ChebyshevI.h

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

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

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



## Public Member Functions

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

## 6.39.1 Detailed Description

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

[Bessel](#) bandstop.

## 6.39.2 Member Function Documentation

6.39.2.1 `setup()`

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

Calculate the coefficients

## Parameters

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

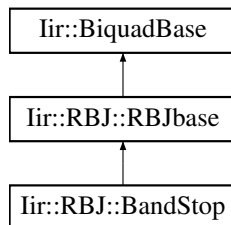
The documentation for this struct was generated from the following file:

- `iir/Bessel.h`

## 6.40 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)

### 6.40.1 Detailed Description

Bandstop

### 6.40.2 Member Function Documentation

#### 6.40.2.1 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 bandpass
<i>bandWidth</i>	Bandwidth of the bandpass

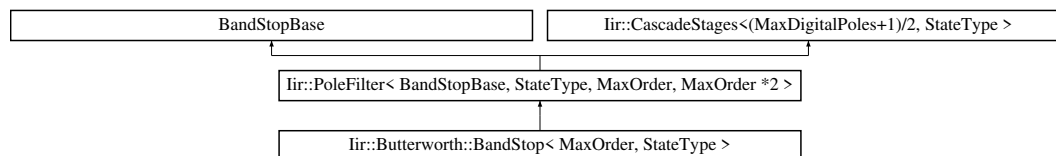
The documentation for this struct was generated from the following files:

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

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

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

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



## Public Member Functions

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

## 6.41.1 Detailed Description

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

[Butterworth](#) Bandstop filter.

## 6.41.2 Member Function Documentation

6.41.2.1 `setup()`

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

Calculates the coefficients

## Parameters

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

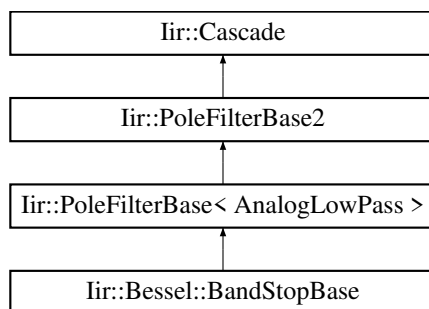
The documentation for this struct was generated from the following file:

- `iir/Butterworth.h`



## 6.42 Iir::Bessel::BandStopBase Struct Reference

Inheritance diagram for Iir::Bessel::BandStopBase:

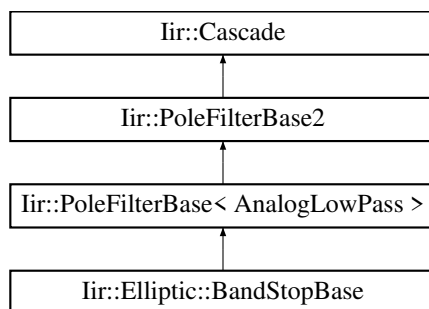


The documentation for this struct was generated from the following files:

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

## 6.43 Iir::Elliptic::BandStopBase Struct Reference

Inheritance diagram for Iir::Elliptic::BandStopBase:

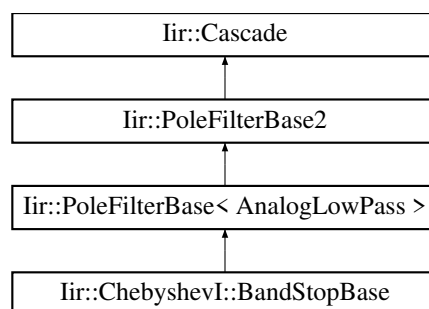


The documentation for this struct was generated from the following files:

- iir/Elliptic.h
- iir/Elliptic.cpp

## 6.44 Iir::ChebyshevI::BandStopBase Struct Reference

Inheritance diagram for Iir::ChebyshevI::BandStopBase:

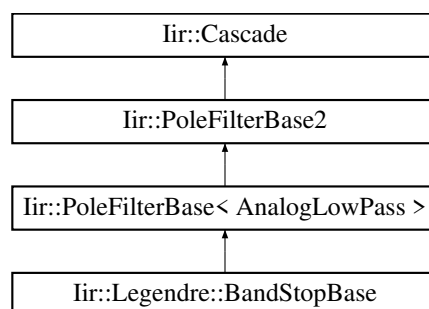


The documentation for this struct was generated from the following files:

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

## 6.45 Iir::Legendre::BandStopBase Struct Reference

Inheritance diagram for Iir::Legendre::BandStopBase:

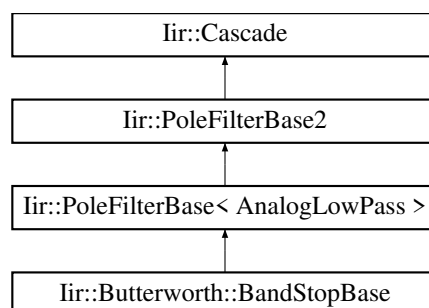


The documentation for this struct was generated from the following files:

- iir/Legendre.h
- iir/Legendre.cpp

## 6.46 Iir::Butterworth::BandStopBase Struct Reference

Inheritance diagram for Iir::Butterworth::BandStopBase:

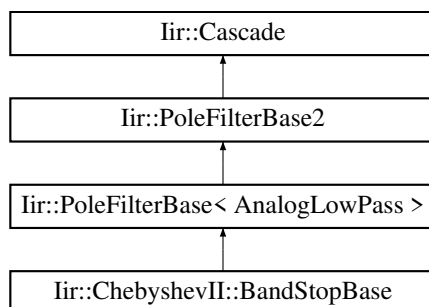


The documentation for this struct was generated from the following files:

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

### 6.47 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

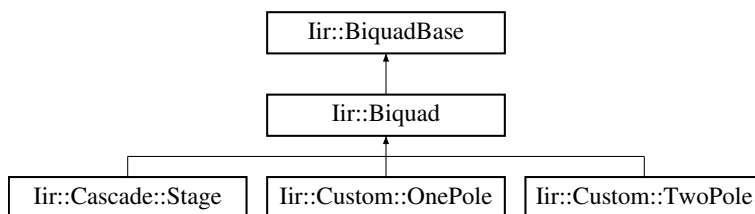
### 6.48 Iir::BandStopTransform Class Reference

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

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

### 6.49 Iir::Biquad Class Reference

Inheritance diagram for Iir::Biquad:

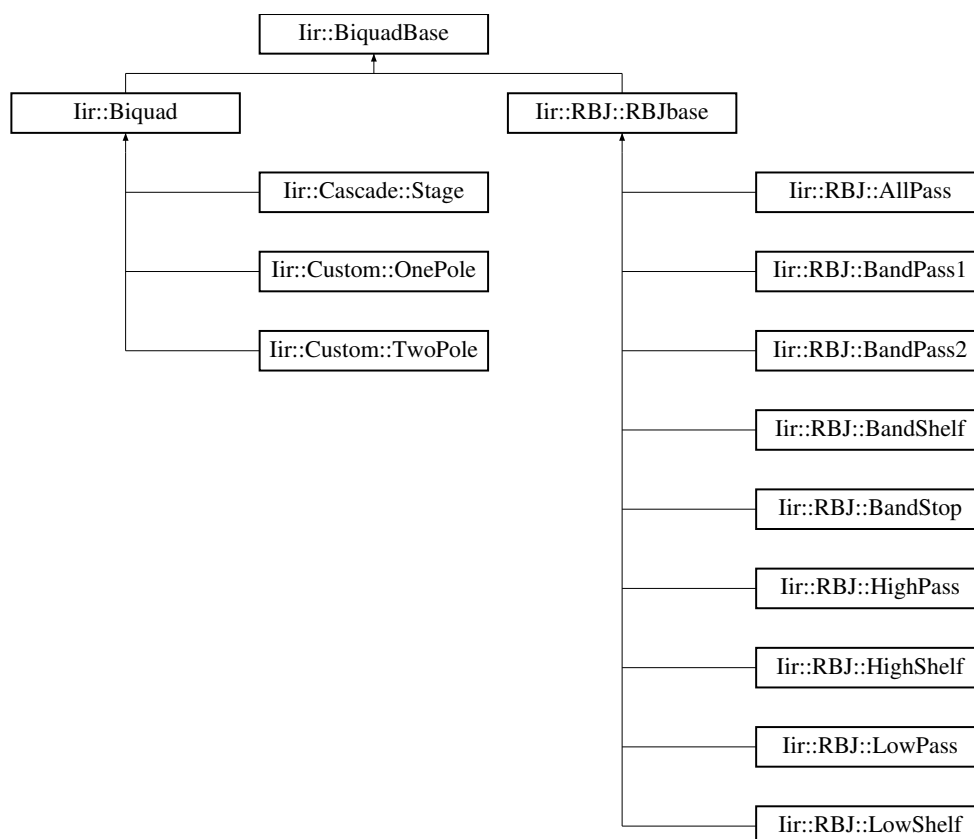


The documentation for this class was generated from the following files:

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

## 6.50 Iir::BiquadBase Class Reference

Inheritance diagram for Iir::BiquadBase:



## Classes

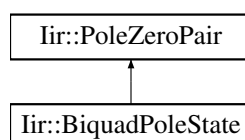
- struct [State](#)

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

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

## 6.51 Iir::BiquadPoleState Struct Reference

Inheritance diagram for Iir::BiquadPoleState:

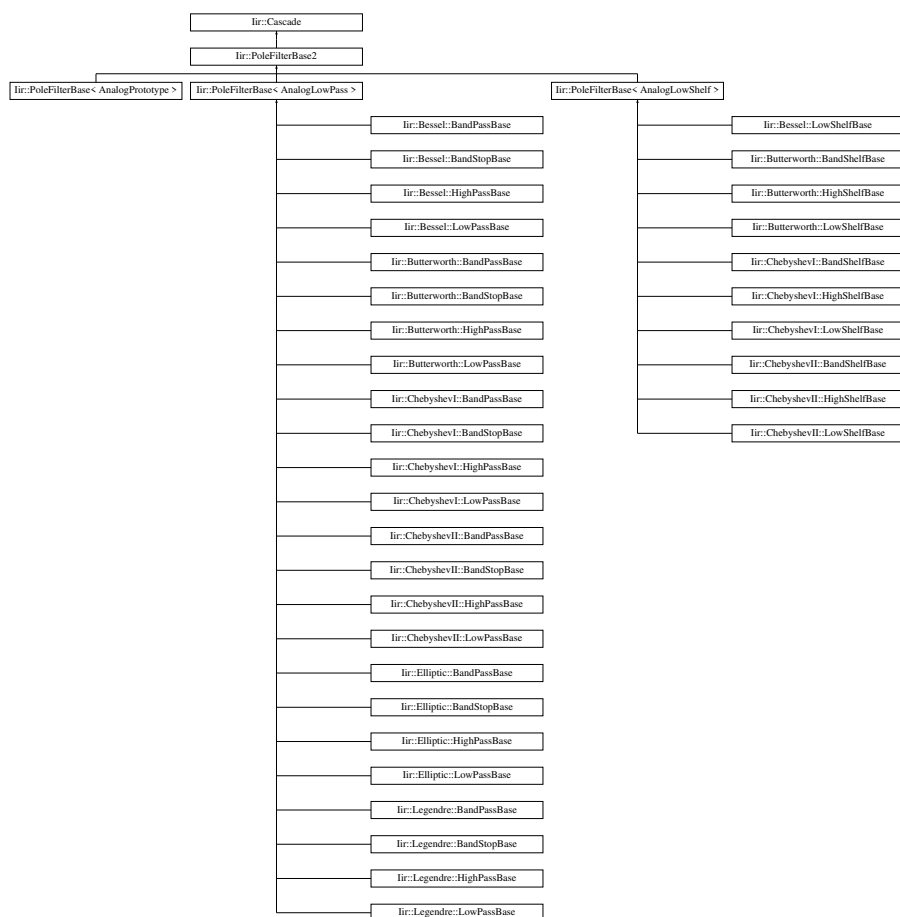


The documentation for this struct was generated from the following files:

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

## 6.52 Iir::Cascade Class Reference

Inheritance diagram for Iir::Cascade:



### Classes

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

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

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

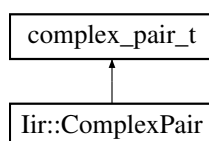
## 6.53 Iir::CascadeStages< MaxStages, StateType > Class Template Reference

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

- iir/Cascade.h

## 6.54 Iir::ComplexPair Struct Reference

Inheritance diagram for Iir::ComplexPair:



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

- iir/Types.h

## 6.55 Iir::DirectFormI Class Reference

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

- iir/State.h

## 6.56 Iir::DirectFormII Class Reference

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

- iir/State.h

## 6.57 Iir::EnvelopeFollower< Channels, Value > Class Template Reference

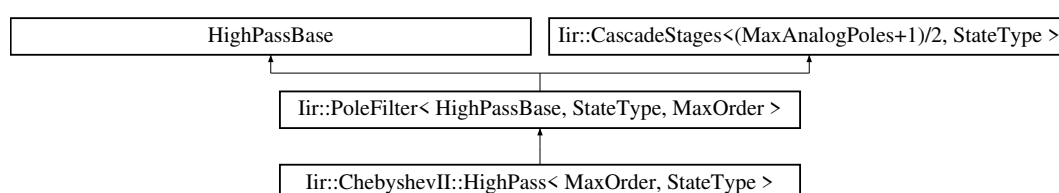
The documentation for this class was generated from the following file:

- iir/Utilities.h

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

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

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



## Public Member Functions

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

## 6.58.1 Detailed Description

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

[ChebyshevII](#) highpass filter

## 6.58.2 Member Function Documentation

## 6.58.2.1 setup()

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

Calculates the coefficients of the filter

## Parameters

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

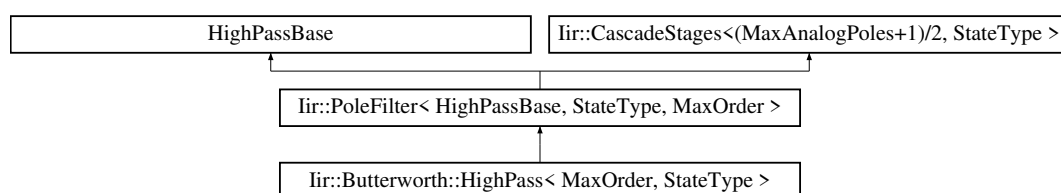
The documentation for this struct was generated from the following file:

- iir/ChebyshevII.h

## 6.59 Iir::Butterworth::HighPass&lt; MaxOrder, StateType &gt; Struct Template Reference

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

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



## Public Member Functions

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

## 6.59.1 Detailed Description

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

[Butterworth](#) Highpass filter.

## 6.59.2 Member Function Documentation

## 6.59.2.1 setup()

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

Calculates the coefficients

## Parameters

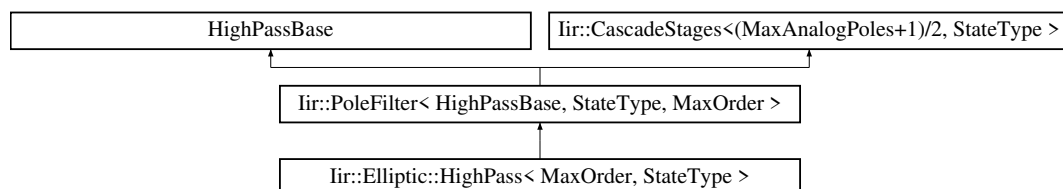
<i>sampleRate</i>	Sampling rate
<i>cutoffFrequency</i>	Cutoff

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

- iir/Butterworth.h

## 6.60 Iir::Elliptic::HighPass&lt; MaxOrder, StateType &gt; Struct Template Reference

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



## Public Member Functions

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



### 6.60.1 Member Function Documentation

#### 6.60.1.1 setup()

```
template<int MaxOrder, class StateType = DEFAULT_STATE>
void Iir::Elliptic::HighPass< MaxOrder, StateType >::setup (
    double sampleRate,
    double cutoffFrequency,
    double rippleDb,
    double rolloff ) [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
<i>rolloff</i>	Rolloff from the pass- to stopband

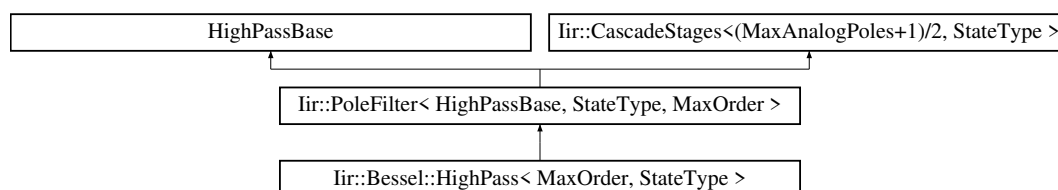
The documentation for this struct was generated from the following file:

- iir/Elliptic.h

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

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

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



#### Public Member Functions

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

#### 6.61.1 Detailed Description

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

[Bessel](#) Highpass.

## 6.61.2 Member Function Documentation

## 6.61.2.1 setup()

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

Calculate the coefficients

## Parameters

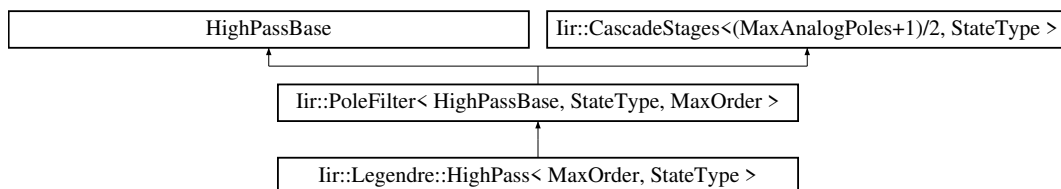
<i>sampleRate</i>	Sampling rate
<i>cutoffFrequency</i>	Cutoff frequency

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

- iir/Bessel.h

## 6.62 Iir::Legendre::HighPass&lt; MaxOrder, StateType &gt; Struct Template Reference

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



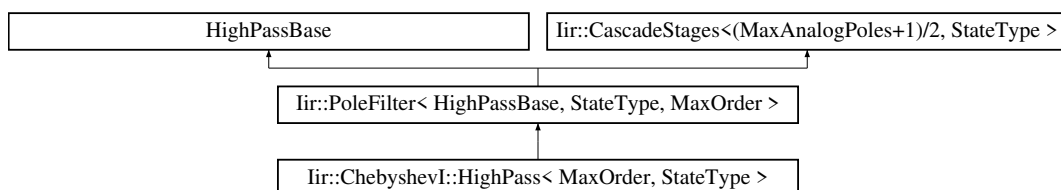
The documentation for this struct was generated from the following file:

- iir/Legendre.h

## 6.63 Iir::ChebyshevI::HighPass&lt; MaxOrder, StateType &gt; Struct Template Reference

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

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



## Public Member Functions

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

### 6.63.1 Detailed Description

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

[ChebyshevI](#) highpass filter

### 6.63.2 Member Function Documentation

#### 6.63.2.1 setup()

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

Calculates the coefficients of the filter

#### Parameters

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

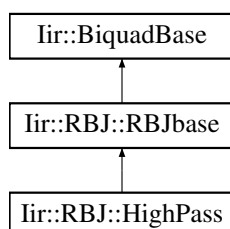
The documentation for this struct was generated from the following file:

- iir/ChebyshevI.h

## 6.64 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)

## 6.64.1 Detailed Description

Highpass.

## 6.64.2 Member Function Documentation

## 6.64.2.1 setup()

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

Calculates the coefficients

## Parameters

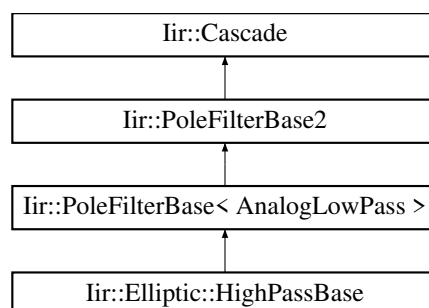
<i>sampleRate</i>	Sampling rate
<i>cutoffFrequency</i>	Cutoff frequency
<i>q</i>	Q factor determines the resonance peak at the cutoff.

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

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

## 6.65 Iir::Elliptic::HighPassBase Struct Reference

Inheritance diagram for Iir::Elliptic::HighPassBase:

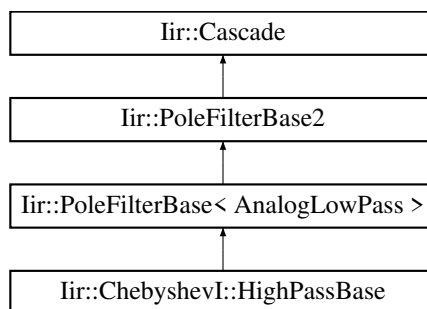


The documentation for this struct was generated from the following files:

- iir/Elliptic.h
- iir/Elliptic.cpp

## 6.66 Iir::ChebyshevI::HighPassBase Struct Reference

Inheritance diagram for Iir::ChebyshevI::HighPassBase:

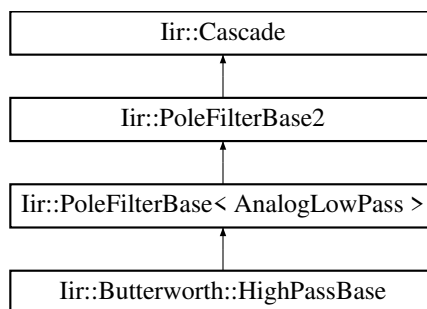


The documentation for this struct was generated from the following files:

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

## 6.67 Iir::Butterworth::HighPassBase Struct Reference

Inheritance diagram for Iir::Butterworth::HighPassBase:

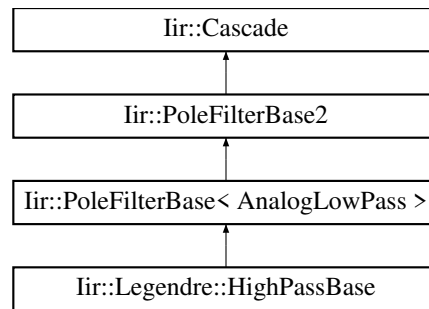


The documentation for this struct was generated from the following files:

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

## 6.68 Iir::Legendre::HighPassBase Struct Reference

Inheritance diagram for Iir::Legendre::HighPassBase:

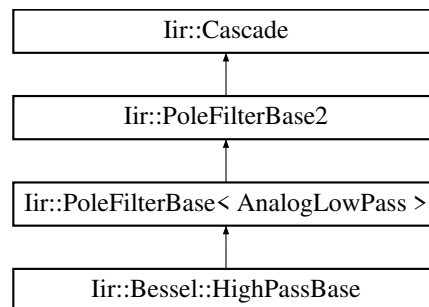


The documentation for this struct was generated from the following files:

- iir/Legendre.h
- iir/Legendre.cpp

## 6.69 Iir::Bessel::HighPassBase Struct Reference

Inheritance diagram for Iir::Bessel::HighPassBase:

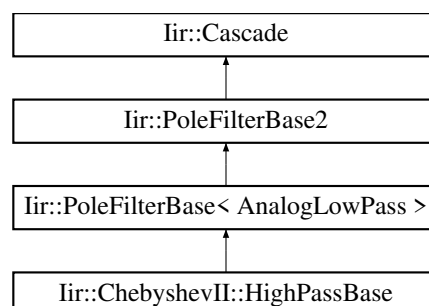


The documentation for this struct was generated from the following files:

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

## 6.70 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

## 6.71 Iir::HighPassTransform Class Reference

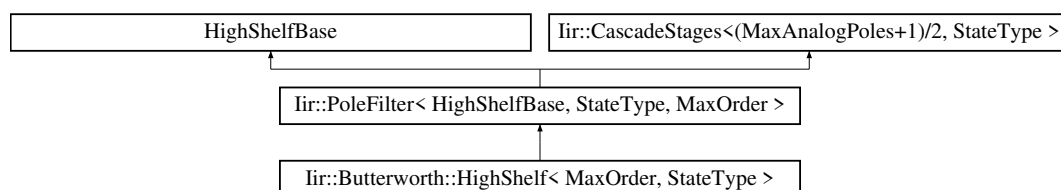
The documentation for this class was generated from the following files:

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

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

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

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



### Public Member Functions

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

### 6.72.1 Detailed Description

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

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

### 6.72.2 Member Function Documentation

#### 6.72.2.1 setup()

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

Calculates the coefficients

## Parameters

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

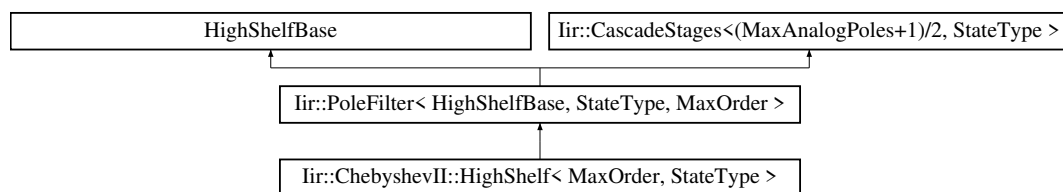
The documentation for this struct was generated from the following file:

- `iir/Butterworth.h`

6.73 `lir::ChebyshevII::HighShelf< MaxOrder, StateType >` Struct Template Reference

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

Inheritance diagram for `lir::ChebyshevII::HighShelf< MaxOrder, StateType >`:



## Public Member Functions

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

## 6.73.1 Detailed Description

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

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

## 6.73.2 Member Function Documentation

6.73.2.1 `setup()`

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

Calculates the coefficients of the filter



## Parameters

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

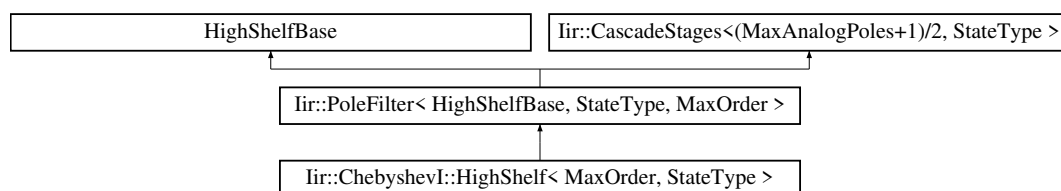
The documentation for this struct was generated from the following file:

- `iir/ChebyshevII.h`

## 6.74 `Iir::ChebyshevI::HighShelf< MaxOrder, StateType >` Struct Template Reference

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

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



## Public Member Functions

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

### 6.74.1 Detailed Description

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

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

### 6.74.2 Member Function Documentation

#### 6.74.2.1 `setup()`

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

Calculates the coefficients of the filter

## Parameters

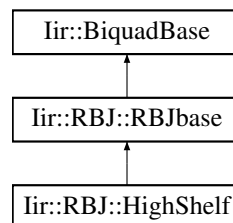
<i>sampleRate</i>	Sampling rate
<i>cutoffFrequency</i>	Cutoff frequency.
<i>gainDb</i>	Gain in the passband
<i>rippleDb</i>	Permitted ripples in dB in the passband

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

- iir/ChebyshevI.h

## 6.75 Iir::RBJ::HighShelf Struct Reference

Inheritance diagram for Iir::RBJ::HighShelf:



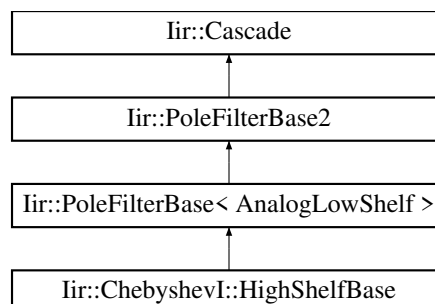
## Additional Inherited Members

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

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

## 6.76 Iir::ChebyshevI::HighShelfBase Struct Reference

Inheritance diagram for Iir::ChebyshevI::HighShelfBase:

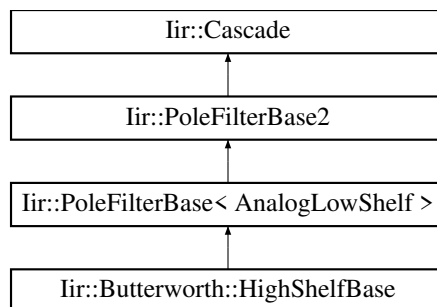


The documentation for this struct was generated from the following files:

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

### 6.77 Iir::Butterworth::HighShelfBase Struct Reference

Inheritance diagram for Iir::Butterworth::HighShelfBase:

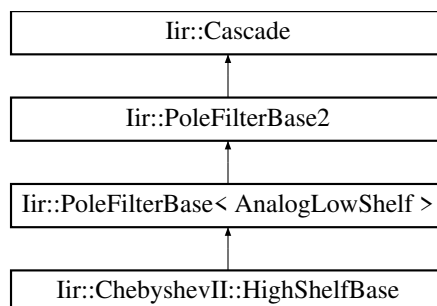


The documentation for this struct was generated from the following files:

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

### 6.78 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

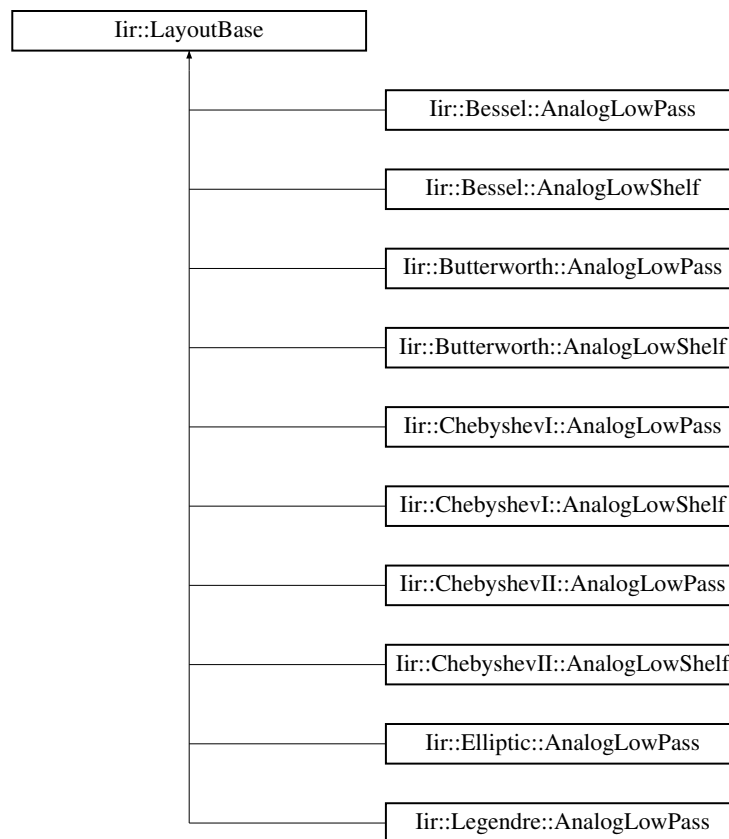
### 6.79 Iir::Layout< MaxPoles > Class Template Reference

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

- iir/Layout.h

## 6.80 Iir::LayoutBase Class Reference

Inheritance diagram for Iir::LayoutBase:



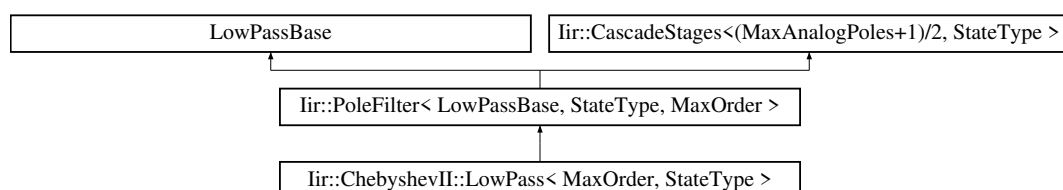
The documentation for this class was generated from the following file:

- iir/Layout.h

## 6.81 Iir::ChebyshevII::LowPass&lt; MaxOrder, StateType &gt; Struct Template Reference

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

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



## Public Member Functions

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

### 6.81.1 Detailed Description

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

[ChebyshevII](#) lowpass filter

### 6.81.2 Member Function Documentation

#### 6.81.2.1 setup()

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

Calculates the coefficients of the filter

#### Parameters

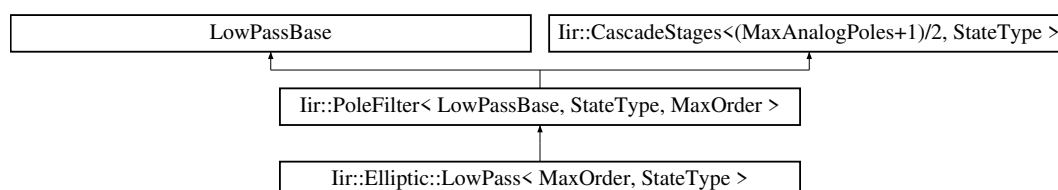
<i>sampleRate</i>	Sampling rate
<i>cutoffFrequency</i>	Cutoff frequency.
<i>stopBandDb</i>	Permitted ripples in dB in the stopband

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

- [iir/ChebyshevII.h](#)

## 6.82 Iir::Elliptic::LowPass< MaxOrder, StateType > Struct Template Reference

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



#### Public Member Functions

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

## 6.82.1 Member Function Documentation

6.82.1.1 `setup()`

```
template<int MaxOrder, class StateType = DEFAULT_STATE>
void Iir::Elliptic::LowPass< MaxOrder, StateType >::setup (
    double sampleRate,
    double cutoffFrequency,
    double rippleDb,
    double rolloff ) [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
<i>rolloff</i>	Rolloff from the pass- to stopband

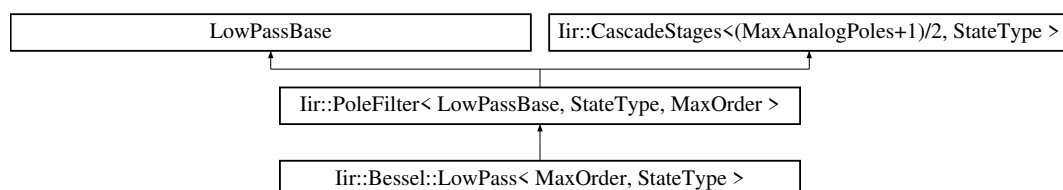
The documentation for this struct was generated from the following file:

- `iir/Elliptic.h`

6.83 `Iir::Bessel::LowPass< MaxOrder, StateType >` Struct Template Reference

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

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



## Public Member Functions

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

## 6.83.1 Detailed Description

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

[Bessel](#) Lowpass

### 6.83.2 Member Function Documentation

#### 6.83.2.1 setup()

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

Calculate the coefficients

#### Parameters

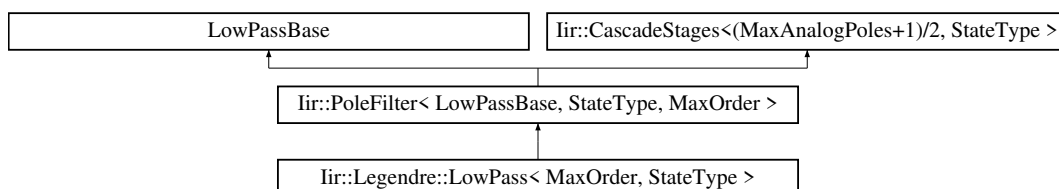
<i>sampleRate</i>	Sampling rate
<i>cutoffFrequency</i>	Cutoff frequency

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

- iir/Bessel.h

### 6.84 Iir::Legendre::LowPass< MaxOrder, StateType > Struct Template Reference

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



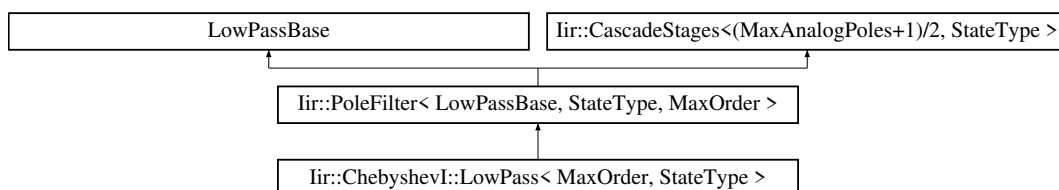
The documentation for this struct was generated from the following file:

- iir/Legendre.h

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

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

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



## Public Member Functions

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

## 6.85.1 Detailed Description

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

[ChebyshevI](#) lowpass filter

## 6.85.2 Member Function Documentation

## 6.85.2.1 setup()

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

Calculates the coefficients of the filter

## Parameters

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

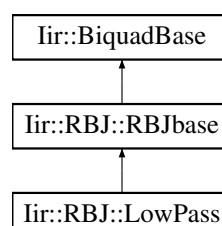
The documentation for this struct was generated from the following file:

- iir/ChebyshevI.h

## 6.86 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)

### 6.86.1 Detailed Description

Lowpass.

### 6.86.2 Member Function Documentation

#### 6.86.2.1 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.

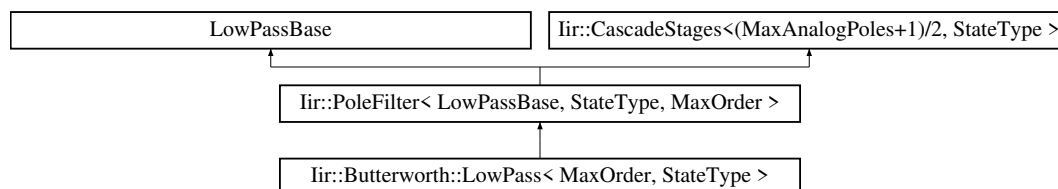
The documentation for this struct was generated from the following files:

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

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

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

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



## Public Member Functions

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

## 6.87.1 Detailed Description

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

[Butterworth](#) Lowpass filter.

## 6.87.2 Member Function Documentation

## 6.87.2.1 setup()

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

Calculates the coefficients

## Parameters

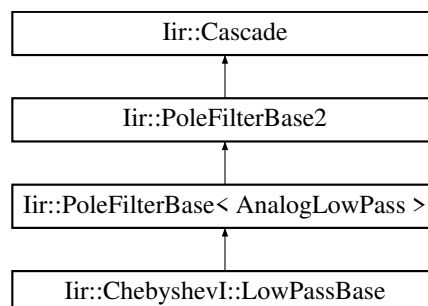
<i>sampleRate</i>	Sampling rate
<i>cutoffFrequency</i>	Cutoff

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

- [iir/Butterworth.h](#)

## 6.88 Iir::ChebyshevI::LowPassBase Struct Reference

Inheritance diagram for Iir::ChebyshevI::LowPassBase:

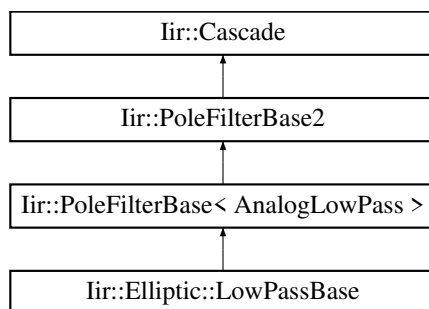


The documentation for this struct was generated from the following files:

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

### 6.89 Iir::Elliptic::LowPassBase Struct Reference

Inheritance diagram for Iir::Elliptic::LowPassBase:

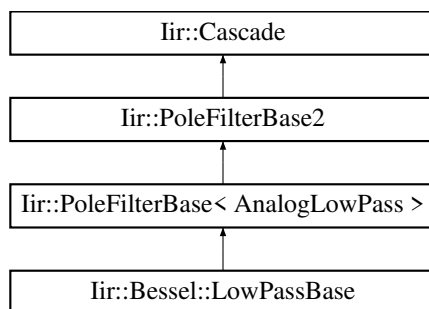


The documentation for this struct was generated from the following files:

- iir/Elliptic.h
- iir/Elliptic.cpp

### 6.90 Iir::Bessel::LowPassBase Struct Reference

Inheritance diagram for Iir::Bessel::LowPassBase:

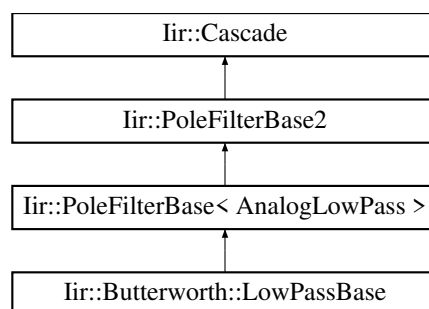


The documentation for this struct was generated from the following files:

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

### 6.91 Iir::Butterworth::LowPassBase Struct Reference

Inheritance diagram for Iir::Butterworth::LowPassBase:

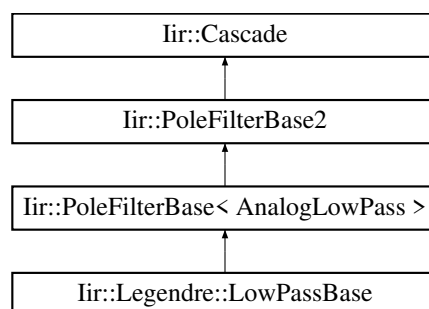


The documentation for this struct was generated from the following files:

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

## 6.92 Iir::Legendre::LowPassBase Struct Reference

Inheritance diagram for Iir::Legendre::LowPassBase:

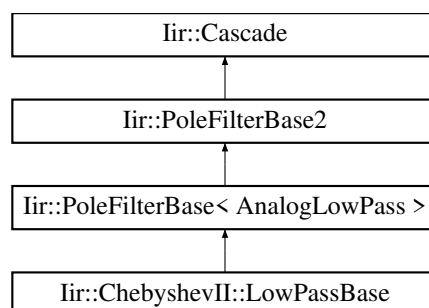


The documentation for this struct was generated from the following files:

- iir/Legendre.h
- iir/Legendre.cpp

## 6.93 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

## 6.94 Iir::LowPassTransform Class Reference

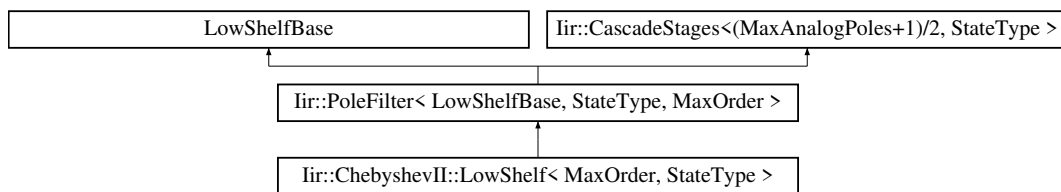
The documentation for this class was generated from the following files:

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

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

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

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



### Public Member Functions

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

### 6.95.1 Detailed Description

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

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

### 6.95.2 Member Function Documentation

#### 6.95.2.1 setup()

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

Calculates the coefficients of the filter

## Parameters

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

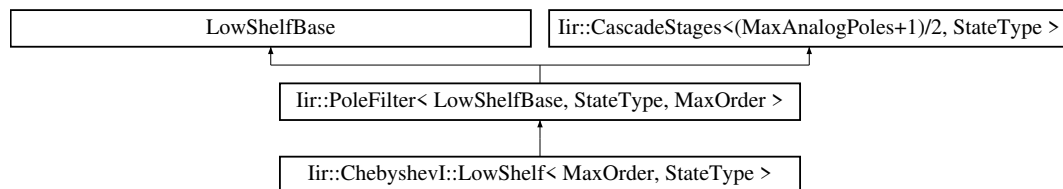
The documentation for this struct was generated from the following file:

- `iir/ChebyshevI.h`

6.96 `Iir::ChebyshevI::LowShelf< MaxOrder, StateType >` Struct Template Reference

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

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



## Public Member Functions

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

## 6.96.1 Detailed Description

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

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

## 6.96.2 Member Function Documentation

6.96.2.1 `setup()`

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

Calculates the coefficients of the filter

**Parameters**

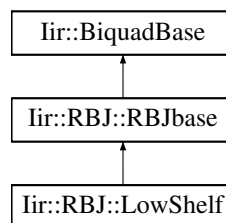
<i>sampleRate</i>	Sampling rate
<i>cutoffFrequency</i>	Cutoff frequency.
<i>gainDb</i>	Gain in the passband
<i>rippleDb</i>	Permitted ripples in dB in the passband

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

- iir/ChebyshevI.h

**6.97 Iir::RBJ::LowShelf Struct Reference**

Inheritance diagram for Iir::RBJ::LowShelf:

**Additional Inherited Members**

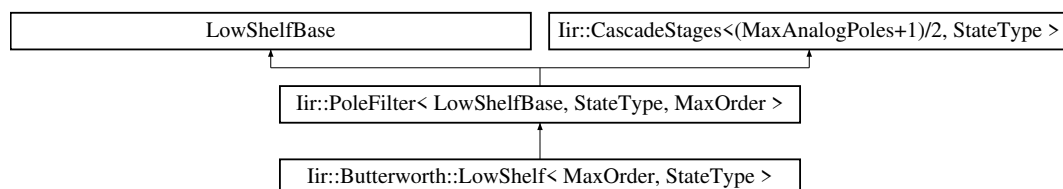
The documentation for this struct was generated from the following files:

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

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

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

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

**Public Member Functions**

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

## 6.98.1 Detailed Description

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

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

## 6.98.2 Member Function Documentation

## 6.98.2.1 setup()

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

Calculates the coefficients

## Parameters

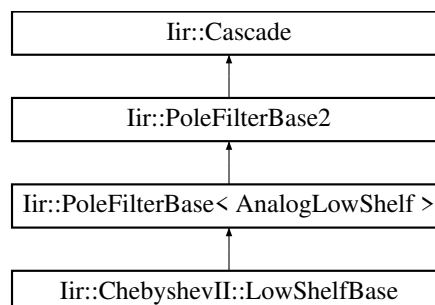
<i>sampleRate</i>	Sampling rate
<i>cutoffFrequency</i>	Cutoff
<i>gainDb</i>	Gain in dB of the filter in the passband

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

- iir/Butterworth.h

## 6.99 Iir::ChebyshevII::LowShelfBase Struct Reference

Inheritance diagram for Iir::ChebyshevII::LowShelfBase:



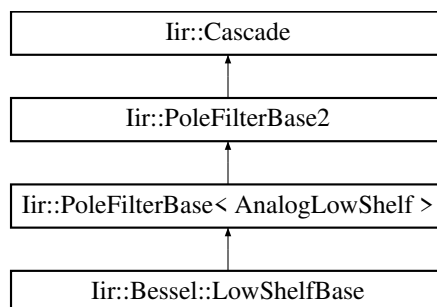
The documentation for this struct was generated from the following files:

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



### 6.100 Iir::Bessel::LowShelfBase Struct Reference

Inheritance diagram for Iir::Bessel::LowShelfBase:

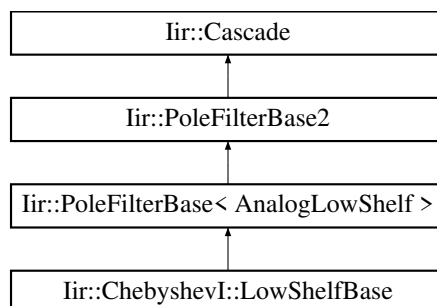


The documentation for this struct was generated from the following files:

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

### 6.101 Iir::ChebyshevI::LowShelfBase Struct Reference

Inheritance diagram for Iir::ChebyshevI::LowShelfBase:

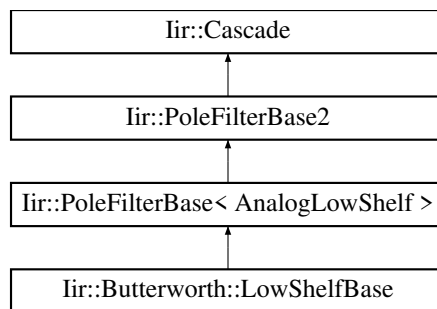


The documentation for this struct was generated from the following files:

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

### 6.102 Iir::Butterworth::LowShelfBase Struct Reference

Inheritance diagram for Iir::Butterworth::LowShelfBase:

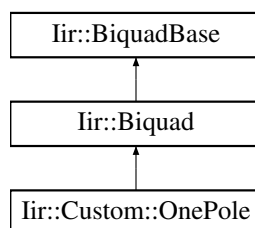


The documentation for this struct was generated from the following files:

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

### 6.103 Iir::Custom::OnePole Struct Reference

Inheritance diagram for Iir::Custom::OnePole:

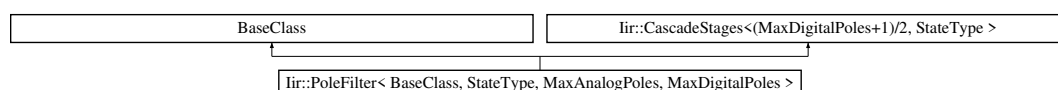


The documentation for this struct was generated from the following files:

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

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

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

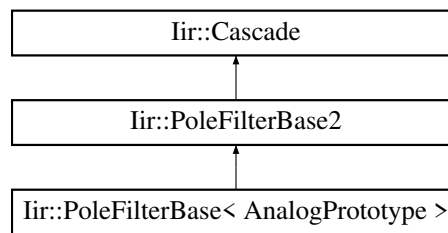


The documentation for this struct was generated from the following file:

- iir/PoleFilter.h

### 6.105 Iir::PoleFilterBase< AnalogPrototype > Class Template Reference

Inheritance diagram for Iir::PoleFilterBase< AnalogPrototype >:

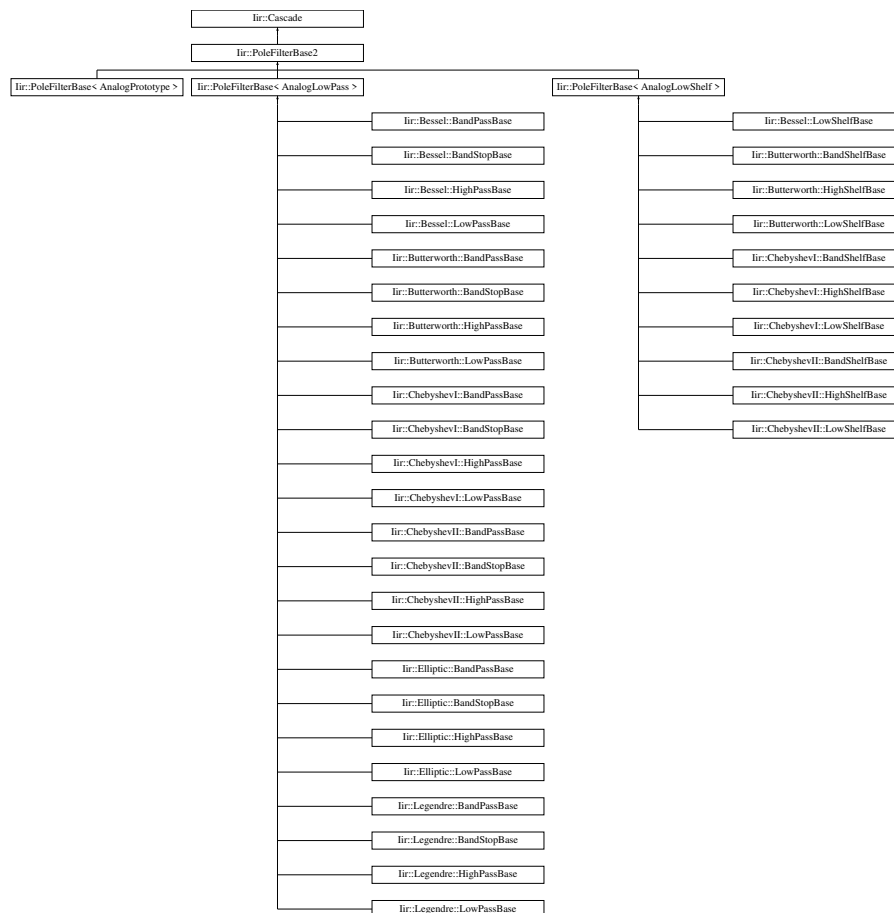


The documentation for this class was generated from the following file:

- iir/PoleFilter.h

### 6.106 Iir::PoleFilterBase2 Class Reference

Inheritance diagram for Iir::PoleFilterBase2:

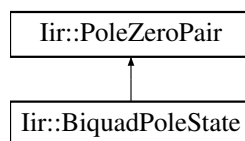


The documentation for this class was generated from the following file:

- iir/PoleFilter.h

## 6.107 Iir::PoleZeroPair Struct Reference

Inheritance diagram for Iir::PoleZeroPair:

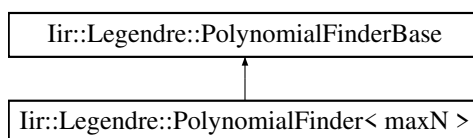


The documentation for this struct was generated from the following file:

- iir/Types.h

## 6.108 Iir::Legendre::PolynomialFinder&lt; maxN &gt; Class Template Reference

Inheritance diagram for Iir::Legendre::PolynomialFinder< maxN >:

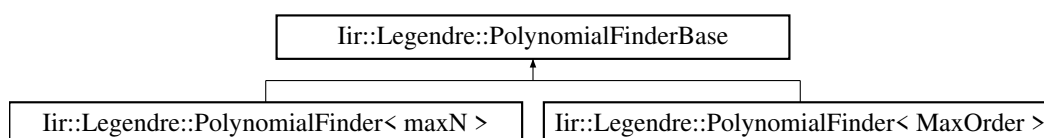


The documentation for this class was generated from the following file:

- iir/Legendre.h

## 6.109 Iir::Legendre::PolynomialFinderBase Class Reference

Inheritance diagram for Iir::Legendre::PolynomialFinderBase:



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

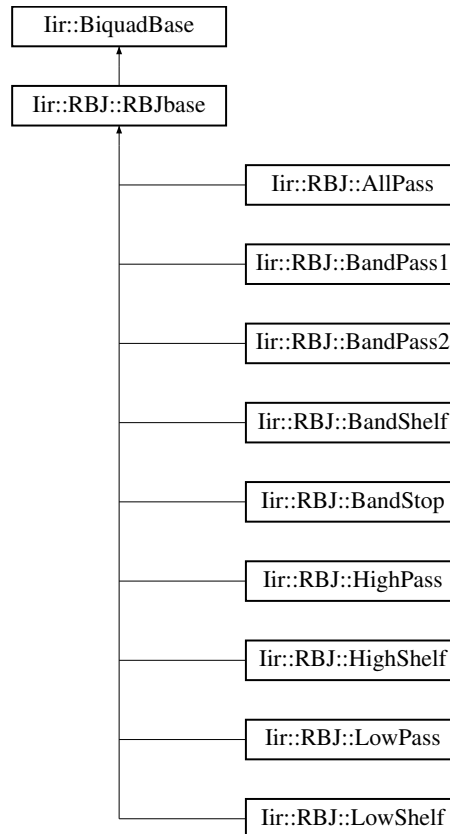
- iir/Legendre.h
- iir/Legendre.cpp

### 6.110 Iir::RBJ::RBJbase Struct Reference

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

```
#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*

#### 6.110.1 Detailed Description

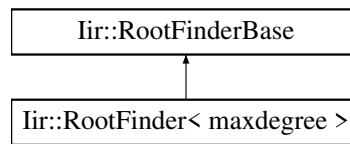
The base class of all [RBJ](#) filters.

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

- `iir/RBJ.h`

## 6.111 Iir::RootFinder&lt; maxdegree &gt; Struct Template Reference

Inheritance diagram for Iir::RootFinder< maxdegree >:

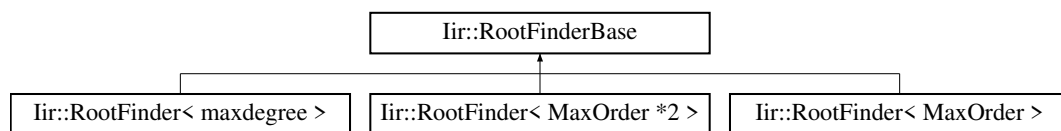


The documentation for this struct was generated from the following file:

- iir/RootFinder.h

## 6.112 Iir::RootFinderBase Class Reference

Inheritance diagram for Iir::RootFinderBase:



## Classes

- struct [Array](#)

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

- iir/RootFinder.h
- iir/RootFinder.cpp

## 6.113 Iir::SlopeDetector&lt; Channels, Value &gt; Class Template Reference

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

- iir/Utilities.h

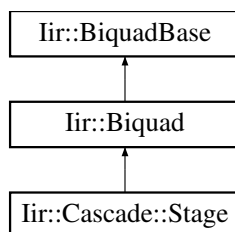
## 6.114 Iir::Elliptic::Solver Class Reference

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

- iir/Elliptic.h
- iir/Elliptic.cpp

### 6.115 Iir::Cascade::Stage Struct Reference

Inheritance diagram for Iir::Cascade::Stage:

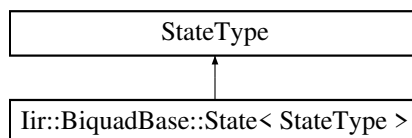


The documentation for this struct was generated from the following file:

- iir/Cascade.h

### 6.116 Iir::BiquadBase::State< StateType > Struct Template Reference

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



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

- iir/Biquad.h

### 6.117 Iir::Cascade::Storage Struct Reference

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

- iir/Cascade.h

### 6.118 Iir::TransposedDirectFormI Class Reference

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

- iir/State.h

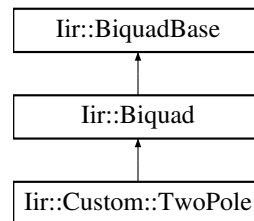
## 6.119 Iir::TransposedDirectFormII Class Reference

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

- iir/State.h

## 6.120 Iir::Custom::TwoPole Struct Reference

Inheritance diagram for Iir::Custom::TwoPole:

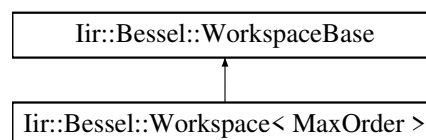


The documentation for this struct was generated from the following files:

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

## 6.121 Iir::Bessel::Workspace< MaxOrder > Struct Template Reference

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

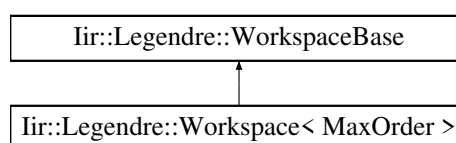


The documentation for this struct was generated from the following file:

- iir/Bessel.h

## 6.122 Iir::Legendre::Workspace< MaxOrder > Struct Template Reference

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



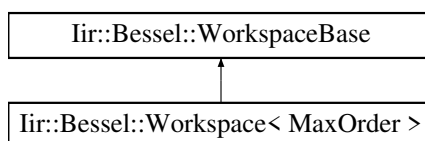
The documentation for this struct was generated from the following file:

- iir/Legendre.h



### 6.123 Iir::Bessel::WorkspaceBase Struct Reference

Inheritance diagram for Iir::Bessel::WorkspaceBase:

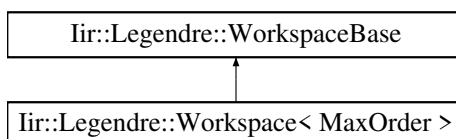


The documentation for this struct was generated from the following file:

- iir/Bessel.h

### 6.124 Iir::Legendre::WorkspaceBase Struct Reference

Inheritance diagram for Iir::Legendre::WorkspaceBase:



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

- iir/Legendre.h

## Index

lir::BandPassTransform, [31](#)  
lir::BandStopTransform, [44](#)  
lir::Bessel, [14](#)  
lir::Bessel::AnalogLowPass, [19](#)  
lir::Bessel::AnalogLowShelf, [20](#)  
lir::Bessel::BandPass  
    setup, [24](#)  
lir::Bessel::BandPass< MaxOrder, StateType >, [23](#)  
lir::Bessel::BandPassBase, [30](#)  
lir::Bessel::BandStop  
    setup, [39](#)  
lir::Bessel::BandStop< MaxOrder, StateType >, [39](#)  
lir::Bessel::BandStopBase, [42](#)  
lir::Bessel::HighPass  
    setup, [51](#)  
lir::Bessel::HighPass< MaxOrder, StateType >, [50](#)  
lir::Bessel::HighPassBase, [55](#)  
lir::Bessel::LowPass  
    setup, [64](#)  
lir::Bessel::LowPass< MaxOrder, StateType >, [63](#)  
lir::Bessel::LowPassBase, [68](#)  
lir::Bessel::LowShelfBase, [74](#)  
lir::Bessel::Workspace< MaxOrder >, [81](#)  
lir::Bessel::WorkspaceBase, [82](#)  
lir::Biquad, [44](#)  
lir::BiquadBase, [45](#)  
lir::BiquadBase::State< StateType >, [80](#)  
lir::BiquadPoleState, [45](#)  
lir::Butterworth, [14](#)  
lir::Butterworth::AnalogLowPass, [18](#)  
lir::Butterworth::AnalogLowShelf, [20](#)  
lir::Butterworth::BandPass  
    setup, [26](#)  
lir::Butterworth::BandPass< MaxOrder, StateType >, [25](#)  
lir::Butterworth::BandPassBase, [29](#)  
lir::Butterworth::BandShelf  
    setup, [31](#)  
lir::Butterworth::BandShelf< MaxOrder, StateType >, [31](#)  
lir::Butterworth::BandShelfBase, [35](#)  
lir::Butterworth::BandStop  
    setup, [41](#)  
lir::Butterworth::BandStop< MaxOrder, StateType >, [41](#)  
lir::Butterworth::BandStopBase, [43](#)  
lir::Butterworth::HighPass  
    setup, [49](#)  
lir::Butterworth::HighPass< MaxOrder, StateType >, [48](#)  
lir::Butterworth::HighPassBase, [54](#)  
lir::Butterworth::HighShelf  
    setup, [56](#)  
lir::Butterworth::HighShelf< MaxOrder, StateType >, [56](#)  
lir::Butterworth::HighShelfBase, [60](#)  
lir::Butterworth::LowPass  
    setup, [67](#)  
lir::Butterworth::LowPass< MaxOrder, StateType >, [66](#)  
lir::Butterworth::LowPassBase, [68](#)  
lir::Butterworth::LowShelf  
    setup, [73](#)  
lir::Butterworth::LowShelf< MaxOrder, StateType >, [72](#)  
lir::Butterworth::LowShelfBase, [74](#)  
lir::Cascade, [46](#)  
lir::Cascade::Stage, [80](#)  
lir::Cascade::Storage, [80](#)  
lir::CascadeStages< MaxStages, StateType >, [46](#)  
lir::ChebyshevI::AnalogLowPass, [18](#)  
lir::ChebyshevI::AnalogLowShelf, [20](#)  
lir::ChebyshevI::BandPass  
    setup, [25](#)  
lir::ChebyshevI::BandPass< MaxOrder, StateType >, [24](#)  
lir::ChebyshevI::BandPassBase, [29](#)  
lir::ChebyshevI::BandShelf  
    setup, [33](#)  
lir::ChebyshevI::BandShelf< MaxOrder, StateType >, [33](#)  
lir::ChebyshevI::BandShelfBase, [34](#)  
lir::ChebyshevI::BandStop  
    setup, [38](#)  
lir::ChebyshevI::BandStop< MaxOrder, StateType >, [38](#)  
lir::ChebyshevI::BandStopBase, [42](#)  
lir::ChebyshevI::HighPass  
    setup, [52](#)  
lir::ChebyshevI::HighPass< MaxOrder, StateType >, [51](#)  
lir::ChebyshevI::HighPassBase, [54](#)  
lir::ChebyshevI::HighShelf  
    setup, [58](#)  
lir::ChebyshevI::HighShelf< MaxOrder, StateType >, [58](#)  
lir::ChebyshevI::HighShelfBase, [59](#)  
lir::ChebyshevI::LowPass  
    setup, [65](#)  
lir::ChebyshevI::LowPass< MaxOrder, StateType >, [64](#)  
lir::ChebyshevI::LowPassBase, [67](#)  
lir::ChebyshevI::LowShelf  
    setup, [71](#)  
lir::ChebyshevI::LowShelf< MaxOrder, StateType >, [71](#)  
lir::ChebyshevI::LowShelfBase, [74](#)  
lir::ChebyshevII::AnalogLowPass, [19](#)  
lir::ChebyshevII::AnalogLowShelf, [21](#)  
lir::ChebyshevII::BandPass  
    setup, [22](#)  
lir::ChebyshevII::BandPass< MaxOrder, StateType >, [21](#)  
lir::ChebyshevII::BandPassBase, [30](#)  
lir::ChebyshevII::BandShelf  
    setup, [32](#)  
lir::ChebyshevII::BandShelf< MaxOrder, StateType >, [32](#)  
lir::ChebyshevII::BandShelfBase, [35](#)  
lir::ChebyshevII::BandStop  
    setup, [36](#)

- lir::ChebyshevII::BandStop< MaxOrder, StateType >, 35
- lir::ChebyshevII::BandStopBase, 44
- lir::ChebyshevII::HighPass
  - setup, 48
- lir::ChebyshevII::HighPass< MaxOrder, StateType >, 47
- lir::ChebyshevII::HighPassBase, 55
- lir::ChebyshevII::HighShelf
  - setup, 57
- lir::ChebyshevII::HighShelf< MaxOrder, StateType >, 57
- lir::ChebyshevII::HighShelfBase, 60
- lir::ChebyshevII::LowPass
  - setup, 62
- lir::ChebyshevII::LowPass< MaxOrder, StateType >, 61
- lir::ChebyshevII::LowPassBase, 69
- lir::ChebyshevII::LowShelf
  - setup, 70
- lir::ChebyshevII::LowShelf< MaxOrder, StateType >, 70
- lir::ChebyshevII::LowShelfBase, 73
- lir::ChebyshevII, 15
- lir::Chebyshevl, 15
- lir::ComplexPair, 47
- lir::Custom::OnePole, 75
- lir::Custom::TwoPole, 81
- lir::DirectFormII, 47
- lir::DirectFormI, 47
- lir::Elliptic, 16
- lir::Elliptic::AnalogLowPass, 18
- lir::Elliptic::BandPass
  - setup, 22
- lir::Elliptic::BandPass< MaxOrder, StateType >, 22
- lir::Elliptic::BandPassBase, 28
- lir::Elliptic::BandStop
  - setup, 37
- lir::Elliptic::BandStop< MaxOrder, StateType >, 36
- lir::Elliptic::BandStopBase, 42
- lir::Elliptic::HighPass
  - setup, 50
- lir::Elliptic::HighPass< MaxOrder, StateType >, 49
- lir::Elliptic::HighPassBase, 53
- lir::Elliptic::LowPass
  - setup, 63
- lir::Elliptic::LowPass< MaxOrder, StateType >, 62
- lir::Elliptic::LowPassBase, 68
- lir::Elliptic::Solver, 79
- lir::EnvelopeFollower< Channels, Value >, 47
- lir::HighPassTransform, 56
- lir::Layout< MaxPoles >, 60
- lir::LayoutBase, 61
- lir::Legendre, 16
- lir::Legendre::AnalogLowPass, 19
- lir::Legendre::BandPass< MaxOrder, StateType >, 23
- lir::Legendre::BandPassBase, 29
- lir::Legendre::BandStop< MaxOrder, StateType >, 37
- lir::Legendre::BandStopBase, 43
- lir::Legendre::HighPass< MaxOrder, StateType >, 51
- lir::Legendre::HighPassBase, 54
- lir::Legendre::LowPass< MaxOrder, StateType >, 64
- lir::Legendre::LowPassBase, 69
- lir::Legendre::PolynomialFinder< maxN >, 77
- lir::Legendre::PolynomialFinderBase, 77
- lir::Legendre::Workspace< MaxOrder >, 81
- lir::Legendre::WorkspaceBase, 82
- lir::LowPassTransform, 70
- lir::PoleFilter< BaseClass, StateType, MaxAnalogPoles, MaxDigitalPoles >, 75
- lir::PoleFilterBase< AnalogPrototype >, 76
- lir::PoleFilterBase2, 76
- lir::PoleZeroPair, 77
- lir::RBJ::AllPass, 17
- lir::RBJ::BandPass1, 26
  - setup, 27
- lir::RBJ::BandPass2, 27
  - setup, 28
- lir::RBJ::BandShelf, 34
- lir::RBJ::BandStop, 40
  - setup, 40
- lir::RBJ::HighPass, 52
  - setup, 53
- lir::RBJ::HighShelf, 59
- lir::RBJ::LowPass, 65
  - setup, 66
- lir::RBJ::LowShelf, 72
- lir::RBJ::RBJbase, 78
- lir::RBJ, 17
- lir::RootFinder< maxdegree >, 79
- lir::RootFinderBase, 79
- lir::RootFinderBase::Array, 21
- lir::SlopeDetector< Channels, Value >, 79
- lir::TransposedDirectFormII, 81
- lir::TransposedDirectFormI, 80
- setup
  - lir::Bessel::BandPass, 24
  - lir::Bessel::BandStop, 39
  - lir::Bessel::HighPass, 51
  - lir::Bessel::LowPass, 64
  - lir::Butterworth::BandPass, 26
  - lir::Butterworth::BandShelf, 31
  - lir::Butterworth::BandStop, 41
  - lir::Butterworth::HighPass, 49
  - lir::Butterworth::HighShelf, 56
  - lir::Butterworth::LowPass, 67
  - lir::Butterworth::LowShelf, 73
  - lir::Chebyshevl::BandPass, 25
  - lir::Chebyshevl::BandShelf, 33
  - lir::Chebyshevl::BandStop, 38
  - lir::Chebyshevl::HighPass, 52
  - lir::Chebyshevl::HighShelf, 58
  - lir::Chebyshevl::LowPass, 65
  - lir::Chebyshevl::LowShelf, 71
  - lir::ChebyshevII::BandPass, 22
  - lir::ChebyshevII::BandShelf, 32
  - lir::ChebyshevII::BandStop, 36
  - lir::ChebyshevII::HighPass, 48

`lir::ChebyshevII::HighShelf`, [57](#)  
`lir::ChebyshevII::LowPass`, [62](#)  
`lir::ChebyshevII::LowShelf`, [70](#)  
`lir::Elliptic::BandPass`, [22](#)  
`lir::Elliptic::BandStop`, [37](#)  
`lir::Elliptic::HighPass`, [50](#)  
`lir::Elliptic::LowPass`, [63](#)  
`lir::RBJ::BandPass1`, [27](#)  
`lir::RBJ::BandPass2`, [28](#)  
`lir::RBJ::BandStop`, [40](#)  
`lir::RBJ::HighPass`, [53](#)  
`lir::RBJ::LowPass`, [66](#)