Filter Design Toolbox Release Notes

Contents

Summary by Version

This table provides quick access to what's new in each version. For clarification, see About Release Notes below.

Version (Release)	New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems	Related Documentation at Web Site
Latest Version V3.4 (R2006a)	Yes Details	Yes Summary	Bug Reports at Web site	Printable Release Notes: PDF V3.4 product documentation
V3.3 (R14SP3)	Yes Details	Yes Summary	Bug Reports at Web site	Printable Release Notes: PDF
V3.2 (R14SP2)	Yes Details	Yes Summary	Bug Reports at Web site	Printable Release Notes: PDF

About Release Notes

Use release notes when upgrading to a newer version to learn about new features and changes, and the potential impact on your existing files and practices. Release notes are also beneficial if you use or support multiple versions

If you are not upgrading from the most recent previous version, review release notes for all interim versions, not just for the version you are installing.

New Features and Changes

These include

- New functionality
- Changes to existing functionality
- Changes to system requirements (complete system requirements for the current version are at the MathWorks Web site)

- Any version compatibility considerations associated with each new feature or change

Version Compatibility Considerations

When a new feature or change introduces a known incompatibility between versions, its description includes a **Compatibility Considerations** subsection that details the impact. For a list of all new features and changes that have compatibility impact, see the Compatibility Summary.

Compatibility issues that become known after the product has been released are added to Bug Reports at the MathWorks Web site. Because bug fixes can sometimes result in incompatibilities, also review fixed bugs in Bug Reports for any compatibility impact.

Fixed Bugs and Known Problems

MathWorks Bug Reports is a user-searchable database of known problems, workarounds, and fixes. The MathWorks updates the Bug Reports database as new problems and resolutions become known, so check it as needed for the latest information.

Access Bug Reports at the MathWorks Web site using your MathWorks Account. If you are not logged in to your MathWorks Account when you link to Bug Reports, you are prompted to log in or create an account. You then can view bug fixes and known problems for R14SP2 and more recent releases.

The Bug Reports database was introduced for R14SP2 and does not include information for prior releases. You can access a list of bug fixes made in prior versions via the links in the summary table.

Related Documentation at Web Site

Printable Release Notes (PDF). You can print release notes from the PDF version, located at the MathWorks Web site. The PDF version does not support links to other documents or to the Web site, such as to Bug Reports. Use the browser-based version of release notes for access to all information.

Product Documentation. At the MathWorks Web site, you can access complete product documentation for the current version and some previous versions, as noted in the summary table.

Version 3.4 (R2006a) Filter Design Toolbox

This table summarizes what's new and changed in Version 3.4 (R2006a):

New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems	Related Documentation at Web Site
Yes Details below	Yes Summary	Bug Reports at Web site	Printable Release Notes: PDF V3.4 product documentation

New features and changes introduced in this version are described here.

- Farrow Filters
- IIR Polyphase Decimators and Interpolators
- Single-Rate Allpass Discrete-time and Multirate Filters
- iirlinphase Method for Designing Linear Phase IIR Filters
- Arbitrary Magnitude and Phase Filter Specification Object
- iirlinphase/elliptic Design for Hilbert Transformers
- CIC Filters Provide Full Precision and Specify All Options
- realizemdl Creates Additional Multirate Polyphase Filters
- Nearest Round Mode for dfilt and mfilt Objects
- Cost Method
- New Online Help for fdesign.structure
- Info Method Updated to Include Filter Measurements
- Measurement Display Changes
- realizemdl Creates Additional Multirate Polyphase Filters
- Filter Design Object Now Called Filter Specification Object in the Documentation

Farrow Filters

The toolbox now provides Farrow filter capability with `farrow`. Using `farrow` you create filters based on the structure and a few options. After you create your filter, various analysis functions, like `cost` and `fvtool`, help you

determine your filter's fitness. `realizemdl` works with Farrow filters to produce blocks for Simulink models as well.

IIR Polyphase Decimators and Interpolators

Now the toolbox provides design tools for IIR polyphase decimators and interpolators using `fdesign.decimator` and `fdesign.interpolator`.

Single-Rate Allpass Discrete-time and Multirate Filters

Eight new filter function enable you to design both single-rate and multirate allpass filters, including wave digital filters.

- `dfilt.allpass`
- `dfilt.wdfallpass`
- `dfilt.cascadeallpass`
- `dfilt.cascadewdfallpass`
- `mfilt.iirdecim`
- `mfilt.iirwdfdecim`
- `mfilt.iirinterp`
- `mfilt.iirwdfinterp`

iirlinphase Method for Designing Linear Phase IIR Filters

The new `iirlinphase` method added in this release designs quasi-linear phase IIR filters from a halfband filter specification objects. Use the form

```
hd = design(d,'iirlinphase');
```

when `d` is a halfband specification object. Returned filter object `hd` is an IIR filter with linear phase in the passband.

Arbitrary Magnitude and Phase Filter Specification Object

The new `arbmagnphase` specification object added in this release designs filters where you define the filter magnitude response and the phase response explicitly. Use the form

```
d = fdesign.arbmagnphase();
```

`d` is a filter specification object where the magnitude and phase responses are specified as a complex frequency response you provide.

iirlinphase/elliptic Design for Hilbert Transformers

When you use `fdesign.hilbert` to create a Hilber transformer specification object, the toolbox provides new `ellip` and `iirlinphase` design methods to implement the filter from the specification object as an elliptic filter or as a quasilinear phase IIR filter.

CIC Filters Provide Full Precision and Specify All Options

CIC filters, such as those created by `fdesign.decimator` and `fdesign.interpolator`, now supports full precision and three word and fraction length modes for the property `FilterInternals`.

- `FullPrecision` mode automatically sets the CIC filter word lengths and fraction lengths to maintain the maximum precision in the filtering process. (new)
- `MinWordLengths` mode lets you set the output word length for the filter.
- `SpecifyWordLengths` mode lets you specify the word lengths for all sections of the filter and for the output. But you cannot set the fraction lengths.
- `SpecifyPrecision` mode lets you set all fraction lengths and word lengths for the filter sections and for the output. (new)

For more information, refer to the reference pages for `fdesign.decimator` and `fdesign.interpolator` in the Filter Design Toolbox documentation.

The following example uses the `SpecifyPrecision` mode. Use a decmation factor of 5 and differential delay equal to 1.

```
d=fdesign.decimator(5,'cic',1) % M=5, D=1.

d =

          MultirateType: 'Decimator'
       DecimationFactor: 5
               Response: 'CIC'
          Specification: 'Fp,Ast'
```

```
           Description: {'Passband Frequency';'Aliasing Attenuation(dB)'}
      DifferentialDelay: 1
    NormalizedFrequency: true
                  Fpass: 0.01
                  Astop: 60

hm=design(d) % Use the default multisection design method.

hm =

          FilterStructure: 'Cascaded Integrator-Comb Decimator'
               Arithmetic: 'fixed'
        DifferentialDelay: 1
         NumberOfSections: 2
         DecimationFactor: 5
         PersistentMemory: false

    InputWordLength: 16
    InputFracLength: 15

    FilterInternals: 'FullPrecision'

hm.FilterInternals='specifyPrecision'

hm =

          FilterStructure: 'Cascaded Integrator-Comb Decimator'
               Arithmetic: 'fixed'
        DifferentialDelay: 1
         NumberOfSections: 2
         DecimationFactor: 5
         PersistentMemory: false

       InputWordLength: 16
       InputFracLength: 15

       FilterInternals: 'SpecifyPrecision'
    SectionWordLengths: [21 21 21 21]
    SectionFracLengths: [15 15 15 15]
      OutputWordLength: 21
      OutputFracLength: 15
```

Nearest Round Mode for dfilt and mfilt Objects

`dfilt` and `mfilt` objects include an additional mode for rounding the results of calculations—`nearest`. Results round to the nearest representable value in the

chosen format. Changing this behavior makes `round` for `dfilt` and `mfilt` objects consistent with `round` in Simulink.

For more information about rounding, refer to `fi` in the Fixed Point Toolbox documentation, since the new rounding modes derive from the `fi` object used by fixed-point filters.

Compatibility Considerations. The new `round` mode behavior is now matches MATLAB `round` as well.

Cost Method

After you create a filter, you can use `cost` to determine the arithimetic cost when you filter data. `cost` returns estimates of the add, multiplies, and other operations that occur when you use the filter.

New Online Help for fdesign.structure

With the addition of more `fdesign` methods and specification objects, the toolbox changes the way you get help about a specific design method—the command-line help is now adaptive, recognizing the object and the design method in the help syntax.

The command-line help adapts to the filter specification object you have and the design method you intend to use, and provides help specifically for that combination of specification and method. For example, if you are designing a highpass filter and plan to use the `butter` design method, here is the new way to get help:

```
d = fdesign.highpass('fst,fp,ast,ap',0.45,0.55,1,60))

designmethods(d)


Design Methods for class fdesign.highpass (Fst,Fp,Ast,Ap):


butter
cheby1
cheby2
ellip
equiripple
ifir
kaiserwin
```

```
help(d,'butter') % New help command syntax with object and method.

DESIGN Design a Butterworth IIR filter.
HD = DESIGN(D, 'butter') designs a Butterworth filter specified by the
FDESIGN object D.

HD = DESIGN(..., 'FilterStructure', STRUCTURE) returns a filter with the
structure STRUCTURE.  STRUCTURE is 'df2sos' by default and can be any of
the following.

'df1sos'
'df2sos'
'df1tsos'
'df2tsos'

HD = DESIGN(..., 'MatchExactly', MATCH) designs a Butterworth filter
and matches the frequency and magnitude specification for the band
MATCH exactly.  The other band will exceed the specification.  MATCH
can be 'stopband' or 'passband' and is 'stopband' by default.

% Example #1 - Compare passband and stopband MatchExactly.
h     = fdesign.highpass('Fst,Fp,Ast,Ap', .7, .9, 60, 1);
Hd    = design(h, 'butter', 'MatchExactly', 'passband');
Hd(2) = design(h, 'butter', 'MatchExactly', 'stopband');

% Compare the passband edges in FVTool.
fvtool(Hd);
axis([.89 .91 -2 0]);
```

Suppose you decide to use an equiripple design method instead. Again, the help command with the specification object `d` and the method `equiripple` provides help for that combination.

```
help (d,'equiripple') % New help command syntax with object and method.

DESIGN Design a Equiripple FIR filter.
HD = DESIGN(D, 'equiripple') designs a Equiripple filter specified by the
FDESIGN object D.

HD = DESIGN(..., 'FilterStructure', STRUCTURE) returns a filter with the
structure STRUCTURE.  STRUCTURE is 'dffir' by default and can be any of
the following.

    'dffir'
    'dffirt'
    'dfsymfir'
```

```
    'dfasymfir'
    'fftfir'

HD = DESIGN(..., 'DensityFactor', DENS) specifies the grid density DENS
used in the optimization.  DENS is 16 by default.

HD = DESIGN(..., 'MinPhase', MPHASE) designs a minimum-phase filter
when MPHASE is TRUE.  MPHASE is FALSE by default.

HD = DESIGN(..., 'MinOrder', 'any') designs a minimum-order filter.
The order of the filter can be even or odd. This is the default.

HD = DESIGN(..., 'MinOrder', 'even') designs an minimum-even-order
filter.

HD = DESIGN(..., 'MinOrder', 'odd') designs an minimum-odd-order filter.

% Example #1 - Design a lowpass Equiripple filter in a transposed
structure.
       h  = fdesign.highpass('Fst,Fp,Ast,Ap');
       Hd = design(h, 'equiripple', 'FilterStructure', 'dffirt');
```

Notice that the content is different for the different methods. This makes is easier for you to know what options apply to any combination of specification object and design method.

Info Method Updated to Include Filter Measurements

When you request information about a filter, the information now includes measurements of the filter characteristics based on the filter specifications. These are the same results that `measure` provides.

Measurement Display Changes

`measure` now shows more information and more specific information for any referred object. Now the display provides full text descriptions of the measured values, such as Sampling Frequency (instead of `Fs`) and Stopband Edge instead of `Fstop`. You should find this a more clear presentation of the filter information.

realizemdl Creates Additional Multirate Polyphase Filters

From the command line, you can use `realizemdl` to create realizations for `firdecim`, `firtdecim`, `firinterp`, and `linearinterp` filters. You can also apply `realizemdl` to the new IIR single-rate and multirate filters:

- `dfilt.allpass`
- `dfilt.wdfallpass`
- `dfilt.cascadeallpass`
- `dfilt.cascadewdfallpass`
- `mfilt.iirdecim`
- `mfilt.iirwdfdecim`
- `mfilt.iirinterp`
- `mfilt.iirwdfinterp`

Filter Design Object Now Called Filter Specification Object in the Documentation

When you use `fdesign.`*`response`*, MATLAB returns an object, usually called `d`, that contains the specifications for a filter design. In the documentation, the returned object is now called a *filter specification object*.

For clarity, we renamed the filter design object to filter specification object, because the object specifies the filter specifications, such as the magnitude response parameters. The specification object is not a filter, but an intermediate step in the filter design process that uses `fdesign.`*`response`* and `design`.

Version 3.3 (R14SP3) Filter Design Toolbox

This table summarizes what's new in Version 3.3 (R14SP3):

New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems	Related Documentation at Web Site
Yes Details below	Yes Summary	Bug Reports at Web site	Printable Release Notes: PDF

New features and changes introduced in this version are

- New Approach for Designing Filters
- New Way to Get Help for Filter Designs
- New Demo Programs to Introduce fdesign Filter Design Approach
- Fdesign Now Provides Arbitrary Magnitude Filter Response Design
- Fdesign Now Provides Hilbert and Differentiator Filter Response Design
- Fdesign Objects Now Use a Default Design Method When Available
- butter and ellip Half-Band Design Methods Added for IIR Fdesign Objects
- limitcycle Method Restored to the Toolbox
- normalizefreq Method Added to the Toolbox
- New measure Method for Filters
- With Fdesign Objects, New Passband Zoom View Option
- With Fdesign Objects, New Filter Specification Mask View Option

New Approach for Designing Filters

To unify and take advantage of the object-based nature of the filters in the toolbox, this release introduces a new design approach for filters using filter design objects and new design methods. In the new process, your filter design tasks begin with identifying the filter response you need for your application.

Here is the new process.

1 Determine the response type for your filter.

2. Choose the appropriate `fdesign.response` method to create a filter specifications object.
3. Select the specifications to use to define your filter object. Here you can select minimum order designs, IIR or FIR designs, or designs that specify the filter order as well as the frequency and magnitude specifications, among many choices.
4. Use `designmethods` to find out which design algorithms apply to your specifications object. Select the design method to use.
5. Use `designopts` with your design object to review the input arguments for your specifications object and your selected design method.
6. Now design your filter using your filter design object, the design method you chose, and the input arguments you require.

The result of this process is a filter object that meets your requirements in response shape or form and design by the method you selected.

Based on three design methods and a new help method, you now design filters starting with the desired response and moving to the final filter. These new methods are:

Method	Description
`design`	Design a filter from the specifications using either a default method or a specified method.
`designmethods`	Find out which design methods apply to your current design object, including dependence on the specifications.
`designopts`	Find out which input arguments apply to your design method and design object.

Here is a short example that demonstrates the new design flow. This `fdesign.lowpass` syntax uses the default response specification `'Fp,Fst,Ap,Ast'`, where `Fp` is the passband edge, `Fst` is the stopband edge, `Ap`

specifies the ripple in the passband, and `Ast` defines the desired stopband attenuation.

```
d = fdesign.lowpass % Select the response.
designmethods(d) % Determine the design methods available.
hd = design(d) % Design the filter using the default method equiripple.

d =

               Response: 'Lowpass'
          Specification: 'Fp,Fst,Ap,Ast'
            Description: {4x1 cell}
    NormalizedFrequency: true
                  Fpass: 0.45
                  Fstop: 0.55
                  Apass: 1
                  Astop: 60



Design Methods for class fdesign.lowpass (Fp,Fst,Ap,Ast):


butter
cheby1
cheby2
ellip
equiripple
ifir
kaiserwin
multistage


hd =

     FilterStructure: 'Direct-Form FIR'
          Arithmetic: 'double'
           Numerator: [1x43 double]
    PersistentMemory: false
```

For more information about a particular design method, use the new help capability with your specifications object and the design method as input arguments to `help`.

This help example gets more information about using the `equiripple` method to design a lowpass filter.

```
help(d,'equiripple') % Get help on using equiripple with your lowpass filter.


 DESIGN Design a Equiripple FIR filter.
    HD = DESIGN(D, 'equiripple') designs a Equiripple filter specified by the
    FDESIGN object H.

    HD = DESIGN(..., 'FilterStructure', STRUCTURE) returns a filter with the
    structure STRUCTURE.  STRUCTURE is 'dffir' by default and can be any of
    the following.
```

```
'dffir'
'dffirt'
'dfsymfir'
'fftfir'

HD = DESIGN(..., 'DensityFactor', DENS) specifies the grid density DENS
used in the optimization.  DENS is 16 by default.

HD = DESIGN(..., 'MinPhase', MPHASE) designs a minimum-phase filter
when MPHASE is TRUE.  MPHASE is FALSE by default.

HD = DESIGN(..., 'MinOrder', 'any') designs a minimum-order filter.
The order of the filter can be even or odd. This is the default.

HD = DESIGN(..., 'MinOrder', 'even') designs an minimum-even-order filter.

HD = DESIGN(..., 'MinOrder', 'odd') designs an minimum-odd-order filter.

HD = DESIGN(..., 'StopbandShape', SHAPE) designs a filter whose stopband
has the shape defined by SHAPE.  SHAPE can be 'flat', '1/f', or 'linear'.
SHAPE is 'flat' by default.

HD = DESIGN(..., 'StopbandDecay', DECAY) specifies the decay to use when
'StopbandShape' is not set to 'flat'.  When the shape is '1/f' this
specifies the power that 1/f is raised.  When shaped is 'linear' this
specifies the slope of the stopband in dB/rad/s.

% Example #1 - Design a lowpass Equiripple filter in a transposed structure.
   h  = fdesign.lowpass('Fp,Fst,Ap,Ast');
   Hd = design(h, 'equiripple', 'FilterStructure', 'dffirt');
```

New Way to Get Help for Filter Designs

Getting help about filter design and filter design methods is now dynamic and depends on the design object and method. When you want help about designing a filter, use `help` with both the filter specifications object and the method to use to design the filter. Here is an example.

```
d = fdesign.bandpass(0.25,0.35,0.55,0.65,50,0.05,50)
designmethods(d)

d =

              Response: 'Bandpass'
         Specification: 'Fst1,Fp1,Fp2,Fst2,Ast1,Ap,Ast2'
           Description: {7x1 cell}
    NormalizedFrequency: true
                 Fstop1: 0.25
                 Fpass1: 0.35
                 Fpass2: 0.55
                 Fstop2: 0.65
                 Astop1: 50
                  Apass: 0.05
                 Astop2: 50
```

```
Design Methods for class fdesign.bandpass (Fst1,Fp1,Fp2,Fst2,Ast1,Ap,Ast2):


butter
cheby1
cheby2
ellip
equiripple
kaiserwin

help(d,'kaiserwin')
 DESIGN Design a Kaiser Windowed FIR filter.
    HD = DESIGN(D, 'kaiserwin') designs a Kaiser Windowed filter specified by the
    FDESIGN object H.

    HD = DESIGN(..., 'FilterStructure', STRUCTURE) returns a filter with the
    structure STRUCTURE.  STRUCTURE is 'dffir' by default and can be any of
    the following.

    'dffir'
    'dffirt'
    'dfsymfir'
    'dfasymfir'
    'fftfir'

    HD = DESIGN(..., 'ScalePassband', SCALE) scales the first passband so
    that it has a magnitude of 0 dB after windowing when SCALE is TRUE.
    SCALE is TRUE by default.

    % Example #1 - Design a bandpass Kaiser Windowed FIR filter.
       h  = fdesign.bandpass('Fst1,Fp1,Fp2,Fst2,Ast1,Ap,Ast2');
       Hd = design(h, 'kaiserwin', 'ScalePassband', false);
```

New Demo Programs to Introduce fdesign Filter Design Approach

For this release, the toolbox adds many new tutorial demos that introduce you to using `fdesign` for your filter design tasks. To access the new demos, enter

```
demos
```

at the Command prompt. When the Help system opens, select `Filter Design ->Tutorial Demos` from the Help Navigator tree in the left pane.

Or use the `demo` command with input arguments:

```
demo('toolbox','filter design')
```

to open the Demos directory showing the Filter Design Toolbox demos.

Fdesign Now Provides Arbitrary Magnitude Filter Response Design

The designs available for `fdesign` now include arbitrary magnitude response filters. You use `fdesign.arbmag` with input arguments to specify a vector of frequency points and response values at those points to define a custom filter response curve.

Fdesign Now Provides Hilbert and Differentiator Filter Response Design

The designs available for `fdesign` now include differentiator and Hilbert magnitude response filters. You use `fdesign.differentiator` or `fdesign.hilbert` with input arguments to specify a differentiator or Hilbert filter design object.

Fdesign Objects Now Use a Default Design Method When Available

`design` now applies a default design method if you do not provide the design method as an input. Usually the default method is `equiripple` for FIR filters and `ellip` for IIR filters.

butter and ellip Half-Band Design Methods Added for IIR Fdesign Objects

For designing IIR halfband filters with `fdesign` and `design`, we added both `butter` and `ellip` to the available design methods.

Added multistage Filter Design Method

In addition to single-stage filters, you can now design multistage filters from lowpass filter design objects by applying the `multistage` method to the object.

For example, after you create a lowpass filter object, use `multistage` to create the filter as a multistage filter.

```
d=fdesign.lowpass(0.25,0.35,0.05,50);
hd = design(d,'multistage')

hd =
```

```
 FilterStructure: Cascade
        Stage(1): Direct-Form FIR Polyphase Decimator
        Stage(2): Direct-Form FIR Polyphase Decimator
        Stage(3): Direct-Form FIR Polyphase Interpolator
        Stage(4): Direct-Form FIR Polyphase Interpolator
PersistentMemory: false
```

limitcycle Method Restored to the Toolbox

The function `limitcycle` is now available to test your fixed-point IIR filters for the limit cycle behavior.

normalizefreq Method Added to the Toolbox

To let you convert your filters to use normalized frequency specifications, rather than absolute frequency, the toolbox adds `normalizefreq` for filter objects.

New measure Method for Filters

A new method, `measure`, lets you measure the response of filters after you design them. `measure` returns the response values at a variety of frequencies in the filter magnitude response.

With Fdesign Objects, New Passband Zoom View Option

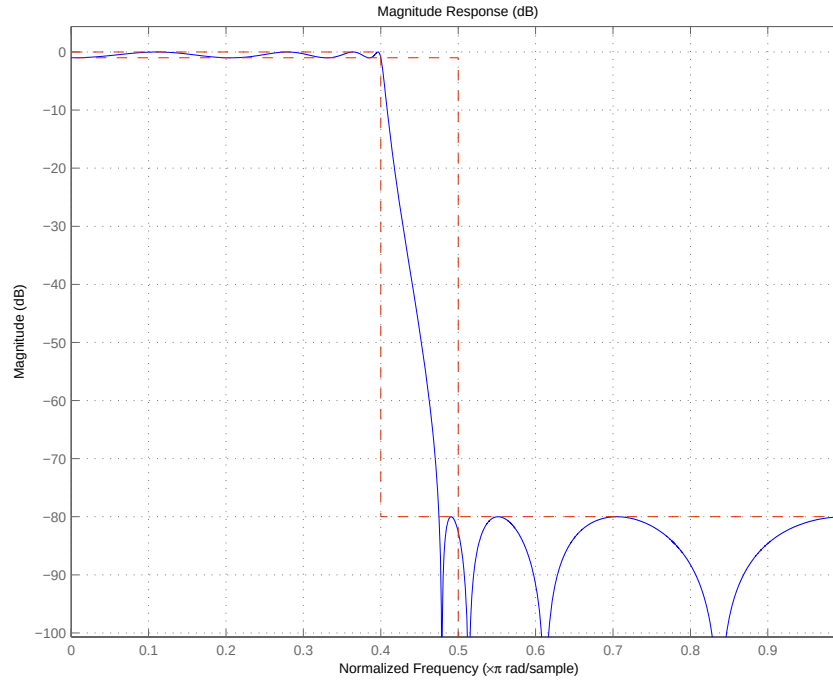
From the **View** menu in FVTool and FDATool, selecting the **Passband** option automatically zooms the display to focus on the passband for the filter shown. Using an `fdesign` object to design your filter enables the Passband Zoom option in FVTool.

With Fdesign Objects, New Filter Specification Mask View Option

When you use FVTool or FDATool to display a filter response for a filter you design with an `fdesign` object, you see new masks that outline the filter passband, stopband, and transition regions as specified by your filter object.

The following graphic shows the mask for a lowpass filter. To display the specification mask, use a filter design object to construct your filter, and then

display the filter in FVTool. select **View->Specification Mask** from the menu bar in FVTool to see the specification mask.



Fdesign Object Display No Longer Shows Fs When the Design Object Uses Normalized Frequency

In this release, the default filter display no longer shows the sampling frequency `Fs` when you specify the filter to use normalized frequency instead of absolute frequency.

For cicinterp Objects, Changed the Order of the Properties in the Display

Reordered the listing of the filter properties in the default display of CIC filters. The new arrangement better matches the display organization for single rate filters.

For IIR Design Objects, Property Fcutoff is Now Called F3dB

The filter property `Fcutoff` is now called `F3dB` to be more descriptive.

Changes to the Displays in MATLAB for Filters

Some of the displays for filter objects, showing the properties and values, are different in this release. Some property names have changed, and some properties reordered to make the displays more logically grouped and consistent across the various objects. Among the changed displays are the CIC object property arrangements and the names of some properties for bandpass, bandstop, and general IIR filter objects.

Compatibility Considerations

If you depend on the displays, be sure to modify your work to accommodate the new display names and arrangements.

Obsolete Functions and Methods in This Release

The following methods are now obsolete.

Compatibility Considerations

As you see in the table, new methods replace them, providing the same or expanded design capability.

Obsolete Method	Replacement Method
`fdesign.decim`	`fdesign.decimator`
`fdesign.interp`	`fdesign.interpolator`
`fdesign.src`	`fdesign.rsrc`

The obsolete methods continue to work, but they may be removed in the future.

block Method for mfilt.firfracdecim Filter Objects No Longer Works

Changes in the FIR Sample Rate Change block in Signal Processing Blockset required that the `block` method for `firfracdecim` filters be made obsolete. You cannot use `block` to create a Simulink block from an `firfracdecim` filter object. To create a `block` from the `firfracdecim` object, convert the object to a `firsrc` object, and then use `block`:

```
hm = mfilt.firfracdecim(4,7);
convert(hm,'firsrc')
block(hm)
```

Compatibility Considerations

Programs that use the `block` method for `firfracdecim` require that you convert your `mfilt.firfracdecim` multirate filter to `firsrc` form using the `convert` method.

Version 3.2 (R14SP2) Filter Design Toolbox

This table summarizes what's new in Version 3.2 (R14SP2):

New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems	Related Documentation at Web Site
Yes Details below	Yes Summary	Bug Reports at Web site	Printable Release Notes: PDF

New features and changes introduced in this version are

- Improved Fixed-Point Support for FIR Filters
- Fixed-Point Linear and Hold Interpolators
- realizemdl Creates CIC Filters
- Context-Sensitive Help for FDATool Returns
- Second-Order Section Filter View Options Available from the Command Line
- Function fdesign Specifies Filter Response with Specified Structure

Improved Fixed-Point Support for FIR Filters

Four FIR filters now support fixed-point processing using the same properties or attributes and methods (mostly) that the fixed-point multirate filters use.

- `dfilt.dfasymfir`
- `dfilt.dffir`
- `dfilt.dffirt`
- `dfilt.dfsymfir`

With the improved filter objects, the properties for your discrete-time filters now look the same as your multirate filters. Unifying the look and feel makes working with the full range of filters in the toolbox easier and more clear.

Additionally, making the switch from floating-point to fixed-point by setting `Arithmetic` to `fixed` creates a fixed-point version of your floating-point filter that uses full precision arithmetic internally. The result—a fixed-point filter that most closesly matches to your floating-point prototype. If your design

cannot support the resources for this fixed-point implementation, you can start to adjust the fixed-point properties as you need.

To continue to use your existing fixed-point FIR filters, you have to upgrade them to the new format. The toolbox includes a new utility for making the transition—`legacyfixptfir`. Note that this utility is not covered in the Filter Design Toolbox documentation. You can get help by entering

```
help legacyfixptfir
```

at the MATLAB prompt.

For information about converting your existing fixed-point FIR filters to the new objects, refer to Upgrading Your Existing Fixed-Point FIR Filters to the New Properties.

Fixed-Point Linear and Hold Interpolators

Both `mfilt.holdinterp` and `mfilt.linearinterp` let you use fixed-point arithmetic. After you create the interpolator object, you can switch the setting for the `Arithmetic` property to `fixed` to use fixed-point interpolation.

Both also support single-precision floating-point arithmetic.

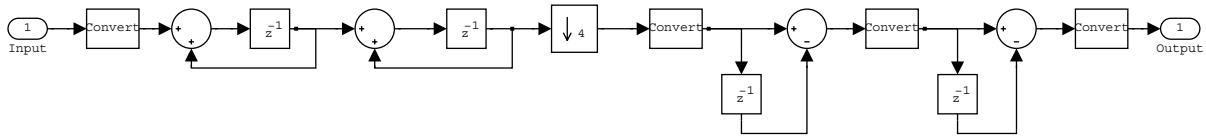
realizemdl Creates CIC Filters

You can use `realizemdl` to construct CIC filters from basic blocks for processing signals. If you construct a CIC decimator filter, as shown here, `realizemdl` can make an atomic subsystem CIC filter block in Simulink for you.

```
hm=mfilt.cicdecim(4);

realizemdl(hm)
```

A new Simulink model window opens and you see a filter block. Double-clicking on the new block shows you the CIC filter subsystem.



Note You must have the Signal Processing Blockset to use `realizemdl` to implement CIC filters.

Context-Sensitive Help for FDATool Returns

FDATool now provides help for options on the quantization, multirate filter design, and frequency transformation panels. Access the new help feature either by right-clicking on an option and selecting **What's This** from the context menu, or clicking the **What's This** help icon on the tool bar.

Second-Order Section Filter View Options Available from the Command Line

In Filter Visualization Tool (FVTool), you can view second-order section filters as "individual sections," "cumulative sections," or as sections that you define. Now this functionality is available from the MATLAB command line, by using the `sosViewSettings` property of the FVTool object. In previous releases these view options were available only as options in the SOS View Settings dialog in FVTool.

Access the FVTool object properties by launching FVTool with a filter object and including a left-hand side output argument:

```
handle = fvtool(hd)
```

`handle` now contains the FVTool properties, similar to the following listing—you use `set` and `get` to manipulate the property values.

```
handle=fvtool(hd)

handle =
```

```
     1

set(handle.sosviewsettings,'view')

ans =

    'Complete'
    'Individual'
    'Cumulative'
    'UserDefined'

set(handle.sosviewsettings,'view','individual')
```

In `SOSViewSettings`, the options are the same, with the same meaning, that you find in **View->SOS View Settings** in FDATool.

For more information about the `fvtool` properties, refer to `fvtool` in the Signal Processing Toolbox documentation or in the online Help system.

Function fdesign Specifies Filter Response with Specified Structure

You can use `fdesign.response` to specify a filter response and specify the filter structure to use during construction.

Upgrading Your Existing Fixed-Point FIR Filters to the New Properties

There is a utility named `legacyfixptfir` to ensure backward compatibility of your existing scripts and a function `update` to help you migrate to the new FIR filters. `legacyfixptfir` switches the preferences for your session between pre- and post-Filter Design Toolbox 3.2 FIR filters.

Here is an example of the process of converting your old FIR filters to the new form in this version of the toolbox.

Begin with an existing direct-form FIR filter `h` that you constructed with

```
h = dfilt.dffir
```

in an earlier version of the toolbox. First, use `legacyfixptfir` to retrieve `h` in the old format. Then convert `h` to the new form.

```
legacyfixptfir(true) % To get the old form of h.
h.Arithmetic='fixed'

h =

     FilterStructure: 'Direct-Form FIR'
          Arithmetic: 'fixed'
           Numerator: 1
    PersistentMemory: false

     CoeffWordLength: 16
      CoeffAutoScale: true
              Signed: true

     InputWordLength: 16
     InputFracLength: 15

    OutputWordLength: 16
          OutputMode: 'AvoidOverflow'

         ProductMode: 'FullPrecision'

           AccumMode: 'KeepMSB'
     AccumWordLength: 40
       CastBeforeSum: true

           RoundMode: 'convergent'
        OverflowMode: 'wrap'

update(h) % Convert h to the new properties.
h

h =

     FilterStructure: 'Direct-Form FIR'
          Arithmetic: 'fixed'
           Numerator: 1
    PersistentMemory: false

     CoeffWordLength: 16
```

```
      CoeffAutoScale: true
              Signed: true

     InputWordLength: 16
     InputFracLength: 15

     FilterInternals: 'SpecifyPrecision'

    OutputWordLength: 16
    OutputFracLength: 13

   ProductWordLength: 32
   ProductFracLength: 29

     AccumWordLength: 40
     AccumFracLength: 29

           RoundMode: 'convergent'
        OverflowMode: 'wrap'
```

Note the changes in properties. The filter performs the same way but the attributes are now updated to the newest form.

Filter Weights Have Been Removed from the Specifications in fdesign

The weights applied to the filter magnitude response are now design options. They are no longer properties of the `fdesign.typeobject`. To set them, pass them as property name/property value (PV) pairs to the appropriate filter design method, as shown in this example.

```
h = fdesign.lowpass('N,Fp,Fst',30) % Was 'N,Fp,Fst,Wp,Wst'.
                                   % Removed Wp and Wst.
hd = equiripple(h, 'Wpass', 3, 'Wstop', 25); % Specify the
                                             % weights here.
hd(2) = equiripple(h, 'Wpass', 3, 'Wstop', 1);
fvtool(hd)
```

Compatibility Summary

This table summarizes new features and changes that might cause incompatibilities when you upgrade from an earlier version, or when you use files on multiple versions of the product. Details about the compatibility effects appear with the description of the new feature or change in the New Features and Changes sections for the product.

Version (Release)	New Features and Changes with Version Compatibility Impact
Latest Version V3.4 (R2006a)	See the **Compatibility Considerations** subheading for "Nearest Round Mode for dfilt and mfilt Objects"
V3.3 (R14SP3)	See the **Compatibility Considerations** subheading for each of these new features or changes: • Fdesign Object Display No Longer Shows Fs When the Design Object Uses Normalized Frequency • Changes to the Displays in MATLAB for Filters • Obsolete Functions and Methods in This Release • block Method for mfilt.firfracdecim Filter Objects No Longer Works
V3.2 (R14SP2)	See the **Compatibility Considerations** subheading for each of these new features or changes: • Upgrading Your Existing Fixed-Point FIR Filters to the New Properties • Filter Weights Have Been Removed from the Specifications in fdesign