Filter Design Toolbox Release Notes

The “Filter Design Toolbox 3.3 Release Notes” on page 1-1 describe the changes introduced in the Filter Design Toolbox 3.3. The following topics are discussed in these release notes:

- “New Features” on page 1-2
- “Major Bug Fixes” on page 1-10
- “Upgrading from an Earlier Release” on page 1-11
- “Known Software and Documentation Problems” on page 1-13

The Filter Design Toolbox Release Notes also provide information about Version 3.2, if you are upgrading from a version that was released prior to Release 14SP2. Refer to the “Filter Design Toolbox 3.2 Release Notes” on page 2-1.

Printing the Release Notes

If you would like to print the release notes, you can link to a PDF version.

Contents

1 Filter Design Toolbox 3.3 Release Notes

New Features 1-2

Major Bug Fixes 1-10

Upgrading from an Earlier Release 1-11

Known Software and Documentation Problems 1-13

2 Filter Design Toolbox 3.2 Release Notes

New Features 2-2

Major Bug Fixes 2-6

Upgrading from an Earlier Release 2-7

Known Software and Documentation Problems 2-10

Filter Design Toolbox 3.3 Release Notes

New Features (p. 1-2)	Provides brief descriptions of the new features in this release
Major Bug Fixes (p. 1-10)	Provides a link to descriptions of the bugs fixed in this release
Upgrading from an Earlier Release (p. 1-11)	Provides important information about changes to existing features in the toolbox
Known Software and Documentation Problems (p. 1-13)	Provides a link to descriptions of the problems identified in this release

New Features

This section introduces the new features and enhancements added to the Filter Design Toolbox 3.3 since Version 3.2.

If you are upgrading from a release earlier than Release 14SP2, refer to "New Features" on page 2-2 in the Filter Design Toolbox 3.2 Release Notes.

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. Use the appropriate `fdesign.response` method to create a filter design object.
3. Select the specifications to use to define your filter object. Here you can minimum order designs, or design that specify the filter order as well as the frequency and magnitude specifications.
4. Use `designmethods` to find out which design techniques apply to your design object. Select the design method to use.
5. Use `designopts` with your design object to review the input arguments for your design 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.

```
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 design 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 design 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.

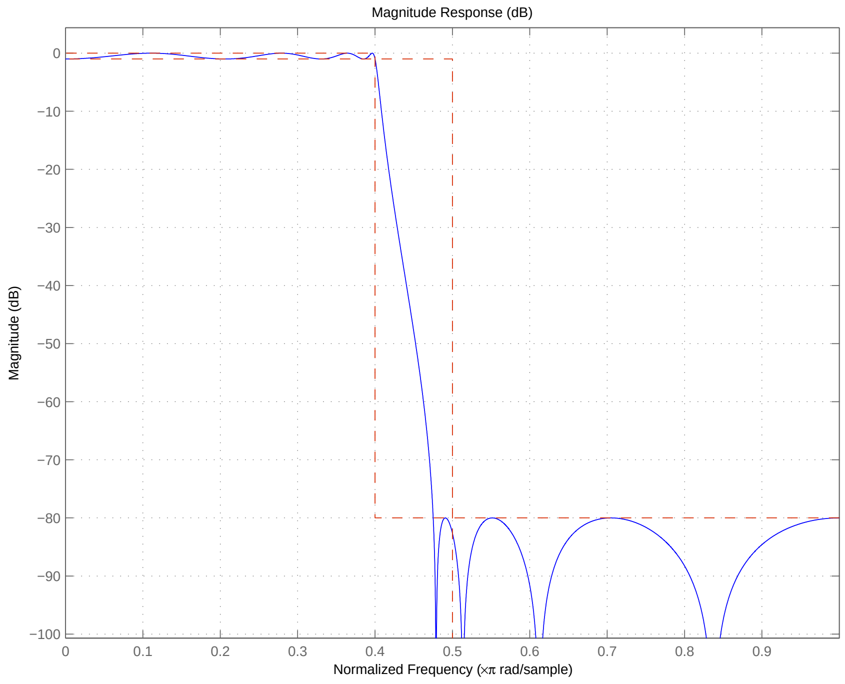
With Fdesign Objects, New Passband Zoom View Option

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.


Magnitude Response (dB)
Magnitude (dB)
0
−10
−20
−30
−40
−50
−60
−70
−80
−90
−100
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
Normalized Frequency (×π rad/sample)

Major Bug Fixes

To view major bug fixes made in Filter Design Toolbox in R14SP3, use the Bug Reports interface on the MathWorks Web site.

Note If you are not already logged in to Access Login, when you link to the Bug Reports interface (see below), you will be prompted to log in or create an Access Login account.

After you are logged in, use this Bug Reports link. You will see the bug report for Filter Design Toolbox. The report is sorted with fixed bugs listed first, and then open bugs.

Upgrading from an Earlier Release

This section describes the upgrade issues involved in moving to the Filter Design Toolbox 3.3 from Version 3.2.

If you are upgrading from a version earlier than Version 3.2 you should also refer to “Upgrading from an Earlier Release” on page 2-7 in the Filter Design Toolbox 3.2 Release Notes.

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.

Obsolete Functions and Methods in This Release

The following methods are now obsolete. 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)
```

Known Software and Documentation Problems

To view important known bugs in Filter Design Toolbox in R14SP3, use the Bug Reports interface on the MathWorks Web site.

Note If you are not already logged in to Access Login, when you link to the Bug Reports interface (see below), you will be prompted to log in or create an Access Login account.

After you are logged in, use this Bug Reports link. You will see the bug report for Filter Design Toolbox. The report is sorted with fixed bugs listed first, and then open bugs.

Filter Design Toolbox 3.2 Release Notes

New Features (p. 2-2)	Provides brief descriptions of the new features in this release
Major Bug Fixes (p. 2-6)	Provides a link to descriptions of the bugs fixed in this release
Upgrading from an Earlier Release (p. 2-7)	Provides important information about changes made to existing features in the toolbox
Upgrading from an Earlier Release (p. 2-7)	Provides a link to descriptions of the problems identified in this release

New Features

This section introduces the new features and enhancements added to the Filter Design Toolbox 3.2 since Version 3.1.

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" on page 2-7.

Fixed-Point Linear and Hold Interpolators

Both `mfilt.holdinterp` and `mfilt.linearinterp` now 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.

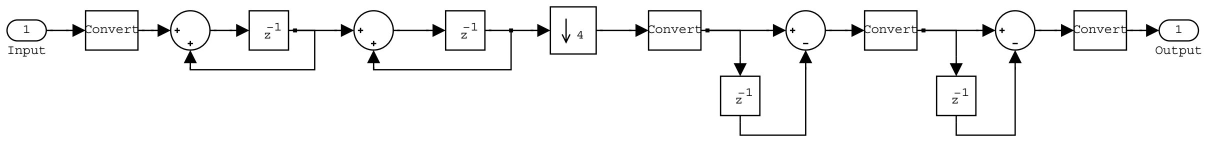
realizemdl Creates CIC Filters

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 Note—you must have the Signal Processing Blockset to use realize model 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 Designs Filters with Specified Structure

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

Major Bug Fixes

To view major bug fixes made in Filter Design Toolbox in R14SP1, use the Bug Reports interface on the MathWorks Web site.

Note If you are not already logged in to Access Login, when you link to the Bug Reports interface (see below), you will be prompted to log in or create an Access Login account.

After you are logged in, use this Bug Reports link. You will see the bug report for Filter Design Toolbox. The report is sorted with fixed bugs listed first, and then open bugs.

Upgrading from an Earlier Release

This section describes the upgrade issues involved in moving to the Filter Design Toolbox 3.2 from Version 3.1.

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)
```

Known Software and Documentation Problems

To view known problems in Filter Design Toolbox in R14SP2, use the Bug Reports interface on the MathWorks Web site.

Note If you are not already logged in to Access Login, when you link to the Bug Reports interface (see below), you will be prompted to log in or create an Access Login account.

After you are logged in, use this Bug Reports link. You will see the bug report for Filter Design Toolbox. The report is sorted with fixed bugs listed first, and then open bugs.