Fixed-Point Toolbox™ Release Notes

How to Contact MathWorks

`www.mathworks.com`	Web
`comp.soft-sys.matlab`	Newsgroup
`www.mathworks.com/contact_TS.html`	Technical Support

`suggest@mathworks.com`	Product enhancement suggestions
`bugs@mathworks.com`	Bug reports
`doc@mathworks.com`	Documentation error reports
`service@mathworks.com`	Order status, license renewals, passcodes
`info@mathworks.com`	Sales, pricing, and general information

508-647-7000 (Phone)

508-647-7001 (Fax)

The MathWorks, Inc.
3 Apple Hill Drive
Natick, MA 01760-2098

For contact information about worldwide offices, see the MathWorks Web site.

Fixed-Point Toolbox™ Release Notes

Trademarks

MATLAB and Simulink are registered trademarks of The MathWorks, Inc. See `www.mathworks.com/trademarks` for a list of additional trademarks. Other product or brand names may be trademarks or registered trademarks of their respective holders.

Patents

MathWorks products are protected by one or more U.S. patents. Please see `www.mathworks.com/patents` for more information.

Contents

Summary by Version

This table provides quick access to what's new in each version. For clarification, see "Using Release Notes" on page 2.

Version (Release)	New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems
Latest Version V3.5 (R2012a)	Yes Details	Yes Summary	Bug Reports Includes fixes
V3.4 (R2011b)	Yes Details	Yes Summary	Bug Reports Includes fixes
V3.3 (R2011a)	Yes Details	No	Bug Reports Includes fixes
V3.2 (R2010b)	Yes Details	Yes Summary	Bug Reports Includes fixes
V3.1 (R2010a)	Yes Details	Yes Summary	Bug Reports Includes fixes
V3.0 (R2009b)	Yes Details	Yes Summary	Bug Reports Includes fixes
V2.4 (R2009a)	Yes Details	Yes Summary	Bug Reports Includes fixes
V2.3 (R2008b)	Yes Details	No	Bug Reports Includes fixes
V2.2.1 (R2008a+)	No	No	Bug Reports Includes fixes

Version (Release)	New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems
V2.2 (R2008a)	Yes Details	No	Bug Reports Includes fixes
V2.1.1 (R2007b+)	No	No	Bug Reports Includes fixes
V2.1 (R2007b)	Yes Details	Yes Summary	Bug Reports Includes fixes
V2.0 (R2007a)	Yes Details	Yes Summary	Bug Reports Includes fixes
V1.5 (R2006b)	Yes Details	No	Bug Reports Includes fixes
V1.4 (R2006a)	Yes Details	Yes Summary	Bug Reports Includes fixes
V1.3 (R14SP3)	Yes Details	Yes Summary	Bug Reports Includes fixes
V1.2 (R14SP2)	Yes Details	No	Bug Reports
V1.1 (R14SP1)	No	No	Yes Details
V1.0 (R14)	Yes Details	Not applicable	No bug fixes

Using Release Notes

Use release notes when upgrading to a newer version to learn about:

- New features
- Changes
- Potential impact on your existing files and practices

Review the release notes for other MathWorks® products required for this product (for example, MATLAB® or Simulink®). Determine if enhancements, bugs, or compatibility considerations in other products impact you.

If you are upgrading from a software version other than the most recent one, review the current release notes and all interim versions. For example, when you upgrade from V1.0 to V1.2, review the release notes for V1.1 and V1.2.

What Is in the Release Notes

New Features and Changes

- New functionality
- Changes to existing functionality

Version Compatibility Considerations

When a new feature or change introduces a reported incompatibility between versions, the **Compatibility Considerations** subsection explains the impact.

Compatibility issues reported after the product release appear under Bug Reports at the MathWorks Web site. Bug fixes can sometimes result in incompatibilities, so review the fixed bugs in Bug Reports for any compatibility impact.

Fixed Bugs and Known Problems

MathWorks offers a user-searchable Bug Reports database so you can view Bug Reports. The development team updates this database at release time and as more information becomes available. Bug Reports include provisions for any known workarounds or file replacements. Information is available for bugs existing in or fixed in Release 14SP2 or later. Information is not available for all bugs in earlier releases.

Access Bug Reports using your MathWorks Account.

Documentation on the MathWorks Web Site

Related documentation is available on `mathworks.com` for the latest release and for previous releases:

- Latest product documentation
- Archived documentation

Version 3.5 (R2012a) Fixed-Point Toolbox Software

This table summarizes what's new in Version 3.5 (R2012a):

New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems
Yes Details below	Yes Summary	Bug Reports Includes fixes

New features and changes introduced in this version are:

Instrumentation Updates

The `showInstrumentationResults` function now has these options:

- `proposeWL` – Proposes word lengths, for fi objects with scaled double data type only
- `proposeFL` – Proposes fraction lengths, for fi objects with scaled double data type only

- `optimizeWholeNumbers` – Optimizes word length of variables that are always whole numbers
- `percentSafetyMargin N` – Sets safety margin of simulation minimum/maximum, where N is a percent
- `browser` – Opens instrumentation results in a web browser window
- `printable` – Creates a printable report

Code Generation for Scaled Double Input

The toolbox now supports code generation for code containing scaled double data types.

Integer Functions Now Return Real-World Values

The following functions now return real-world values instead of stored integer values: `int8`, `int16`, `int32`, `int64`, `uint8`, `uint16`, `uint32`, and `uint64`.

Compatibility Considerations

In code generation with MATLAB Coder™ or Simulink Coder, if you used a Code Replacement Library (CRL) to replace a cast in your replacement function, silent incorrect numerical results may occur. The numerical results will not change if the input fi object has binary-point scaling and zero fractional length. To optimize code generation, these integer functions now use floor rounding, instead of nearest rounding, when the input fraction length equals 0. You should reevaluate your integer cast replacement functions and update their replacement tables.

New fi Stored Integer Functions

`storedInteger` returns the stored integer value of the input fi object in the smallest built-in integer data type.

`storedIntegerToDouble` returns the stored integer value of a fi object in a double-precision, floating-point data type.

fixed.Quantizer Slope-Bias Support

`fixed.Quantizer` now accepts binary-point scaled or slope-bias scaled fixed-point data as input. You can also obtain slope-bias scaled output quantization by specifying the `numerictype` slope and bias.

Autoscaling Enhancement

This release enhances autoscaling so that overflows no longer occur with some edge cases. For example, previously, when constructing this fi object, an overflow occurred:

```
fi(-128.125,1,8,'RoundMode','floor','OverFlowMode','wrap')
ans =

   127    % Overflow

          DataTypeMode: Fixed-point: binary point scaling
            Signedness: Signed
            WordLength: 8
        FractionLength: 0
```

Now the same code does not overflow:

```
fi(-128.125,1,8,'RoundMode','floor','OverFlowMode','wrap')

ans =

  -130

          DataTypeMode: Fixed-point: binary point scaling
            Signedness: Signed
            WordLength: 8
        FractionLength: -1
```

Compatibility Considerations

Data types may change, which may produce numeric differences to avoid overflows. To retain the old result, add the fraction length to the fi call: `fi(-128.125,1,8,0,'RoundMode','floor','OverFlowMode','wrap')`

fixed.Quantizer, accumneg, and accumpos Property Changes

The `RoundingMethod` property now also accepts these settings: `Ceiling`, `Convergent`, `Floor`, `Nearest`, `Round`, and `Zero`. All of these property settings are case insensitive and the old property values continue to work.

fi, fimath, and numerictype Property Changes

In a future release, all `fi`, `fimath`, and `numerictype` property names will be case sensitive and will require that you use the full property names. You will not be able to use truncated property names.

Compatibility Considerations

To avoid seeing warnings in the future, update your code so it uses the full names and correct cases of all `fi`, `fimath` and `numerictype` properties. The full name of correct case of the properties are shown when you display a `fi`, `fimath` or `numerictype` object on the MATLAB command line.

Plus and Minus Operators Enhancements for Code Generation

To enable code generation, the behavior of the plus (+) and minus (-) operators will change in a future release. Double or single data type operands will be converted to fi objects. This may cause overflows or precision loss.

For example, if `x` is a fi object and `xc` is a double or single operand,

```
x + xc
```

will change to behavior equivalent to

```
x + fi(xc, numericyype(x))
```

Compatibility Considerations

To retain the current behavior, update your code by replacing

```
x + xc
```

with

```
x + fi(xc,get(x,'Signed'),get(x,'WordLength')
```

Dynamic Memory Allocation Based on Size for Code Generation

By default, dynamic memory allocation is now enabled for variable-size arrays whose size exceeds a configurable threshold. This behavior allows for finer control over stack memory usage. Also, you can generate code automatically for more MATLAB algorithms without modifying the original MATLAB code.

Compatibility Considerations

If you use scripts to generate code and you do not want to use dynamic memory allocation, you must disable it. See `coder.MexConfig` for information.

Conversion of Error and Warning Message Identifiers

For R2012a, error and warning message identifiers have changed in Fixed-Point Toolbox™ product.

Compatibility Considerations

If you have scripts or functions that use message identifiers that changed, you must update the code to use the new identifiers. Typically, message identifiers are used to turn off specific warning messages, or in code that uses a try/catch statement and performs an action based on a specific error identifier. For example, the `fi:overflow` identifier has changed to `fixed:fi:overflow` and `fi:underflow` has changed to `fixed:fi:underflow`. If your code checks for `fi:overflow` or`fi:underflow`, you must update your code to check for `fixed:fi:overflow` or `fixed:fi:underflow`, respectively, instead.

To determine the identifier for a warning, run the following command just after you see the warning:

```
[MSG,MSGID] = lastwarn;
```

This command saves the message identifier to the variable `MSGID`.

To determine the identifier for an error, run the following command just after you see the error:

```
exception = MException.last;
MSGID = exception.identifier;
```

Note Warning messages indicate a potential issue with your code. While you can turn off a warning, a suggested alternative is to change your code so it does not generate warnings.

New demo

- Setting Fixed-Point Data Types Using Min/Max Instrumentation demonstrates using minimum and maximum values in the instrumentation report to set fixed-point data types.

Functions Being Removed

Function Name	What Happens When You Use the Function?	Use This Instead	Compatibility Considerations
`fixed.accumpos` `fixed.accumneg`	Still runs Still runs	`accumpos` `accumneg`	Update any code that uses the old `fixed.accumpos` or `fixed.accumneg` names.

`globalfimath`
`removedefaultfimathpref`
`removeglobalfimathpref`
`resetdefaultfimath`
`resetglobalfimath`
`savedefaultfimathpref`
	Warns
	Warns
	Warns
	Warns
	Warns
	Warns
		The default fimath values	Using global fimath affects code portability. If you change any global fimath settings from their default values, a warning occurs. Use the

Function Name	What Happens When You Use the Function?	Use This Instead	Compatibility Considerations
`saveglobalfimathpref` `setdefaultfimath`	Warns Warns		default fimath values. See the Fixed-Point Arithmetic demo for more information.

`int`	Warns	`storedInteger`
		or
		`storedIntegerToDouble`
			Update any code that uses the old `int` function to the new `storedInteger` or `storedIntegerToDouble` function.

Version 3.4 (R2011b) Fixed-Point Toolbox Software

This table summarizes what's new in Version 3.4 (R2011b):

New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems
Yes Details below	Yes Summary	Bug Reports Includes fixes

New features and changes introduced in this version are:

Support for fi in MATLAB Compiler

You can now compile code containing fixed-point `fi` objects using MATLAB Compiler™ software.

New Instrumentation Functions

Three new instrumentation functions, plus a demo, that log minimum and maximum values have been added to the toolbox. Using these functions, you can view the minimum and maximum fixed-point values in MATLAB code

simulations. You can use these values to determine the word and fraction lengths for your fixed-point values. The new functions are:

- `buildInstrumentedMex` – Generate a MEX function with logging instrumentation enabled
- `showInstrumentationResults` – Show the results of an instrumented MEX function
- `clearInstrumentationResults` – Clear the results of an instrumented MEX function

New Quantizer Object

A new `fixed.Quantizer` object that converts one `fi` to another `fi` has been added to the toolbox. It reduces or increases a quantity's word length and fraction length.

Fixed-Point Input Support Added to MATLAB Functions

The following MATLAB functions now support fixed-point `fi` inputs:

- `atan2` – Arctangent
- `cos` – Cosine
- `mod` – Modulus after division
- `qr` – Orthogonal-triangular decomposition
- `sin` – Sine

These functions are accurate to within the top 16 most significant bits of the input(s).

New CORDIC-Based Fixed-Point Functions

Four new CORDIC-based, fixed-point functions have been added to the toolbox:

- `cordicabs` – Complex absolute value
- `cordicangle` – Complex angle

- `cordicatan2` – Arctangent
- `cordiccart2pol` – Cartesian to polar conversion

min and max Allow Different Fixed-Point Input Attributes

The input arguments to `min` and `max`, which you use with `fi` objects, can be fixed-point objects, integers, floating-point, or any combination of these data types. The inputs can also have different signedness, word lengths, and fraction lengths.

fimath mpy Allows Double, Single, and Integer Inputs

The elementwise multiplication function, `mpy`, which operates on `fimath`, allows inputs to be doubles, singles, or integers. In earlier releases only `fi` inputs were allowed.

New Accumulator Functions

Two new accumulator functions, `fixed.accumpos` and `fixed.accumneg`, have been added to the toolbox. These functions accumulate without causing the word length to increase. The `fixed.accumpos` function implements += behavior for fixed-point values. The `fixed.accumneg` function implements -= behavior for fixed-point values.

New Aggregation Function

The new `fixed.aggregateType` function returns the aggregate numerictype of two fixed-point inputs. The *aggregate* is the smallest binary point scaled numerictype that can represent both the full range and full precision of the inputs' numerictypes. The inputs must be integers, binary point scaled fixed-point `fi` objects or numerictype objects.

New demos

Two new demos have been added:

- Instrumentation of a Fixed-Point Filter demonstrates using the new instrumentation logging functions.

- Fixed-Point Arithmetic demonstrates basic fixed-point arithmetic operations.

Conversion of Error and Warning Message Identifiers

For R2011b, error and warning message identifiers have changed in Fixed-Point Toolbox software.

Compatibility Considerations

If you have scripts or functions that use message identifiers that changed, you must update the code to use the new identifiers. Typically, message identifiers are used to turn off specific warning messages. You can also use them in code that uses a try/catch statement and performs an action based on a specific error identifier.

For example, the `fi:constructor:invalidInput` identifier has changed to `fixed:fi:invalidConstructorInput`. If your code checks for `fi:constructor:invalidInput`, you must update it to check for `fixed:fi:invalidConstructorInput` instead.

To determine the identifier for a warning, run the following command just after you see the warning:

```
[MSG,MSGID] = lastwarn;
```

This command saves the message identifier to the variable *MSGID*.

To determine the identifier for an error, run the following command just after you see the error:

```
exception = MException.last;
MSGID = exception.identifier;
```

Note Warning messages indicate a potential issue with your code. While you can turn off a warning, a suggested alternative is to change your code so it does not generate warnings.

Functions Being Removed

Function Name	What Happens When You Use the Function?	Use This Instead	Compatibility Considerations

`globalfimath`	Still runs	`fixed.Quantizer`
		`fixed.accumpos`
		`fixed.accumneg`
			Using `globalfimath` affects code portability. Use the default `globalfimath` values, or if you are using non-default values for `globalfimath`, use
			• `fixed.Quantizer`
			• `fixed.accumpos`
			• `fixed.accumneg`
			to implement the desired behavior. See the Fixed-Point Arithmetic demo for an example.

Version 3.3 (R2011a) Fixed-Point Toolbox Software

This table summarizes what's new in Version 3.3 (R2011a):

New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems
Yes Details below	No	Bug Reports Includes fixes

New features and changes introduced in this version are:

New CORDIC Functions

Two new CORDIC functions, `cordicpol2cart` and `cordicrotate`, have been added to the toolbox. `cordicpol2cart` converts polar coordinates to Cartesian coordinates. `cordicrotate` rotates a complex input. Each of these functions uses efficient CORDIC-based approximations to perform the conversion and rotation, respectively.

CORDIC Functions Allow Unsigned Theta

All CORDIC functions now allow the data type of `theta` to be either signed or unsigned. This change expands the usability of these functions.

CORDIC Number of Iterations Argument Now Optional

The number of iterations (`niters`) input argument is now an optional input to all CORDIC functions. If you do not specify the number of iterations or if you specify a value that is too large for the word length, the maximum reasonable number of iterations is used automatically.

New fiaccel Function to Accelerate Fixed-Point Code

A new `fiaccel` function has been added to the toolbox. `fiaccel` generates a MEX function and accelerates fixed-point code. These associated functions have also been added: `coder.ArrayType`, `coder.Constant`, `coder.EnumType`, `coder.FiType`, `coder.newtype`, `coder.PrimitiveType`, `coder.resize`, `coder.StructType`, `coder.Type`, and `coder.typeof`.

New Signedness Syntax Shortcut

The following new signedness shortcuts have been added to the `numerictype` object syntax.

- `[] = 'Signedness','Auto'`
- `0 = 'Signedness','Unsigned'`
- `1 = 'Signedness','Signed'`

For example, `numerictype([],24,20)` is a `numerictype` object with auto signedness, a word length of 24, and a fraction length of 20.

New DataTypeOverride Property Added to numerictype Object

A new `DataTypeOverride` property has been added to the `numerictype` object. The `numerictype` `DataTypeOverride` property controls how `fipref` data type settings are applied to `fi` objects. This property allows you to conveniently ignore a global `fipref` setting or to turn fixed-point settings on or off without having to change every `fi` object.

If you set the `numerictype` `DataTypeOverride` property to `'off'`, the `fi` object uses the `numerictype` data type settings and ignores the `fipref` settings.

If you set the `numerictype` `DataTypeOverride` property to `Inherit`, which is the default, the `fi` object uses the `fipref` `DataTypeOverride` setting. Valid settings are `ForceOffScaledDoubles`, `TrueDoubles`, and `TrueSingles`.

By default, this property is not visible until you create a `numerictype` object and then explicitly set the desired value. See example in the "New DataTypeOverrideAppliesTo Property Added to fipref Object" on page 19 section.

New DataTypeOverrideAppliesTo Property Added to fipref Object

When you set the `fipref` `DataTypeOverride` property to any value other than `ForceOff`, the new `DataTypeOverrideAppliesTo` displays. `DataTypeOverrideAppliesTo` controls the data types to which the `fipref` `DataTypeOverride` property applies. Valid values of `DataTypeOverrideAppliesTo` are `AllNumericTypes`, `Fixed-point`, and `Floating-point`.

By default, this property is not visible until you create a `numerictype` or `fipref` object and then explicitly set the desired values. For example,

```
T = numerictype
P = fipref

% Change DataTypeOverride from its default
T.DataTypeOverride = 'Off';

% Set fipref DataTypeOverride and
%   DataTypeOverrideAppliesTo
P.DataTypeOverride = 'TrueDoubles';
P.DataTypeOverrideAppliesTo = 'FixedPoint;
```

Improved Display Updates for the NumericTypeScope

R2011a introduces the capability to improve the performance of the `NumericTypeScope` by reducing the frequency with which the display updates. You can now choose between this new enhanced performance mode and the

old behavior. By default, the scope operates in the new enhanced performance mode.

Version 3.2 (R2010b) Fixed-Point Toolbox Software

This table summarizes what's new in Version 3.2 (R2010b):

New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems
Yes Details below	Yes Summary	Bug Reports Includes fixes

New features and changes introduced in this version are:

Enhanced NumericTypeScope for Selecting Fixed-Point Data Types

The Fixed-Point Toolbox `NumericTypeScope` has been enhanced for R2010b. The scope now recommends data types based on the criteria you specify in the tool. The scope also allows you to switch into an interactive mode where you can adjust the word and fraction lengths of the data type. When you are working in the interactive mode, you can drag the vertical bars that indicate the representable range of the data type.

Input Angle Range Extended for CORDIC Trigonometric Functions

The following CORDIC trigonometric functions now accept input angles in the range [–2π 2π):

- `cordiccexp`
- `cordiccos`
- `cordicsin`
- `cordicsincos`

Floating-Point Support Added for bitsll Function

The Fixed-Point Toolbox `bitsll` function now supports floating-point inputs. You can now use the `bitsll` function with fixed-point, built-in or floating-point data types.

Enhanced Compilation Report for emlc and emlmex

The **Variables** tab of the compilation report now has an **Order** column. This new column allows you to sort the display of the variables according to the order in which they appear in your code.

Changes to the DataTypeMode Property of the numerictype Object

The capitalization of the following `DataTypeMode` property values has changed for R2010b:

R2010a Property Value	R2010b Property Value
`boolean`	`Boolean`
`double`	`Double`
`single`	`Single`

Compatibility Consideration

As of R2010b, the `DataTypeMode` property of the `numerictype` object returns the property values `Boolean`, `Double` and `Single` with the first letter

capitalized. If your code performs any case-sensitive operations on the `DataTypeMode` property of a `numerictype` object, you must update that code.

If you query the `DataTypeMode` property of a `numerictype` object in R2010b, MATLAB throws a warning. To workaround this warning, try using one or more of the following `numerictype` object functions:

- `isboolean`
- `isdouble`
- `isfixed`
- `isscaleddouble`
- `isscalingbinarypoint`
- `isscalingslopebias`
- `isscalingunspecified`
- `issingle`

New fi and numerictype Object Functions

R2010b adds the following `fi` and `numerictype` object functions:

- `isscalingbinarypoint` — Determine whether input has binary point scaling
- `isscalingslopebias` — Determine whether input has nontrivial slope and bias scaling
- `isscalingunspecified` — Determine whether input has unspecified scaling

These functions provide you with the ability to query the scaling of a `fi` or `numerictype` object.

New and Updated Demos

This release adds the Data Type Independent Code for CORDIC QR Factorization demo. The demo provides an example of how to write your algorithms using data type independent code. By writing your algorithms

using data type independent code, you can run both floating- and fixed-point simulations using the same algorithm.

The Fixed-Point Algorithm Development demo has been updated to use the enhanced `NumericTypeScope` object.

fi Objects with Boolean Data Type No Longer Obey Data Type Override

In previous releases, `fi` objects with a boolean data type participated in data type override. When you set the `DataTypeOverride` property of the `fipref` object to `ScaledDoubles`, `TrueDoubles`, or `TrueSingles`, `fi` objects with a boolean data type were overridden with the specified data type.

As of R2010b, `fi` objects with a boolean data type no longer participate in data type override. `fi` objects with a `DataType` of `boolean` (or a `DataTypeMode` of `Boolean`) now remain boolean `fi` objects, regardless of the current `DataTypeOverride` setting.

Functions Being Removed

Function Name	What Happens When You Use the Function?	Use This Instead	Compatibility Considerations
`removedefaultfimathpref`	Still runs	`removeglobalfimathpref`	Replace all instances of `removedefaultfimathpref` with `removeglobalfimathpref`.
`resetdefaultfimath`	Still runs	`resetglobalfimath`	Replace all instances of `resetdefaultfimath` with `resetglobalfimath`.

Function Name	What Happens When You Use the Function?	Use This Instead	Compatibility Considerations
`savedefaultfimathpref`	Still runs	`saveglobalfimathpref`	Replace all instances of `savedefaultfimathpref` with `saveglobalfimathpref`.
`setdefaultfimath`	Still runs	`globalfimath`	Replace all instances of `setdefaultfimath` with `globalfimath`.

Version 3.1 (R2010a) Fixed-Point Toolbox Software

This table summarizes what's new in Version 3.1 (R2010a):

New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems
Yes Details below	Yes Summary	Bug Reports Includes fixes

New features and changes introduced in this version are:

New Handle Object for the Global fimath

You can now configure the global fimath and return a handle object to it using the `globalfimath` function. Doing so makes it easier to change the properties of the global fimath, whether you are working at the MATLAB command line or in a file. See the "Functions Being Removed" on page 29 release note for information about the function that `globalfimath` replaces.

New Global fimath Functions

The following global fimath functions are new in R2010a:

- `removeglobalfimathpref`
- `resetglobalfimath`
- `saveglobalfimathpref`

For information about the functions they replace, see the "Functions Being Removed" on page 29 release note.

New NumericTypeScope for Selecting Fixed-Point Data Types

Fixed-Point Toolbox software now offers a `NumericTypeScope` object that can perform a dynamic range analysis on your data. You can use the results of the dynamic range analysis to help you select appropriate `numerictype` properties for your data.

New CORDIC Trigonometric Functions

Fixed-Point Toolbox software now offers the following CORDIC-based approximation functions:

- `cordiccexp`
- `cordiccos`
- `cordicsin`
- `cordicsincos`

These functions allow you to compute the sine, cosine, and complex exponential of fixed-point data using the CORDIC approximation method.

Fixed-Point Support Added for filter, mean, and median Functions

Fixed-Point Toolbox software now provides support for the following functions:

- `filter`
- `mean`
- `median`

Fixed-Point Support Added for mpower (^) and power (.^) Functions

Fixed-Point Toolbox software now provides support for the `mpower` (^) and `power` (.^) functions.

CastBeforeSum Property Hidden on fimath Objects with a SumMode of FullPrecision

The setting of the `CastBeforeSum` property does not affect full-precision sums, so effective this release, it is hidden for `fimath` objects that have a `SumMode` of `FullPrecision`. In previous releases, you could not generate code with the Embedded MATLAB® subset when the `CastBeforeSum` property of a `fimath` object was set to `False`, even if you were only computing full-precision sums.

This change ensures that you can always generate code for full-precision sums using `emlmex`, `emlc`, or the Embedded MATLAB Function block. A `CastBeforeSum` property setting of `False` no longer causes an error if you are working with full-precision sums.

New and Updated Demos

This release adds the Fixed-Point Sine and Cosine Calculation demo. This demo shows you how to compute the sine and cosine of fixed-point data using the new CORDIC-based approximation functions.

Demos with significant updates this release include the following:

- Fixed-Point Basics — This demo now highlights the functionality of recently added features, including the new handle object for the global fimath.
- Fixed-Point Algorithm Development — This demo now uses the `NumericTypeScope` object to help select appropriate fixed-point data types.

Functions Being Removed

Function Name	What Happens When You Use the Function?	Use This Instead	Compatibility Considerations
`removedefaultfimathpref`	Still runs	`removeglobalfimathpref`	Replace all instances of `removedefaultfimathpref` with `removeglobalfimathpref`.
`resetdefaultfimath`	Still runs	`resetglobalfimath`	Replace all instances of `resetdefaultfimath` with `resetglobalfimath`.
`savedefaultfimathpref`	Still runs	`saveglobalfimathpref`	Replace all instances of `savedefaultfimathpref` with `saveglobalfimathpref`.
`setdefaultfimath`	Still runs	`globalfimath`	Replace all instances of `setdefaultfimath` with `globalfimath`.

Version 3.0 (R2009b) Fixed-Point Toolbox Software

This table summarizes what's new in Version 3.0 (R2009b):

New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems
Yes Details below	Yes Summary	Bug Reports Includes fixes

New features and changes introduced in this version are:

New Relationship Between fi Objects and a Global fimath Simplifies Fixed-Point Arithmetic

In previous versions of Fixed-Point Toolbox software, all `fi` objects had an attached `fimath` object property. The attached `fimath` object determined how the `fi` object performed fixed-point arithmetic operations. This format made it more cumbersome to create `fi` objects, sometimes resulting in "mismatched `fimath`" errors.

In R2009b, the behavior of the `fi` object has been changed. A `fi` object no longer needs to have its own attached `fimath` object property. Rather than each `fi` object having its own set of `fimath` object properties, you can now associate `fi` objects with the global fimath. See Working with the Global fimath section for more information.

When you create a `fi` object in R2009b without specifying any `fimath` object properties in the constructor, MATLAB returns a `fi` object that associates itself with the global fimath. If you change the global fimath, all `fi` objects that are associated with the global fimath automatically pick up the new `fimath` properties.

You can still use the `fi` object constructor function to specify a particular `fimath` object for a `fi` object.

In general, fixed-point arithmetic operations use the following rules:

- If either `fi` object in a binary operation has its own `fimath` object, the operation uses that `fimath` object.
- If both `fi` objects in a binary operation use the global fimath, the operation uses the global fimath.

For more information on fixed-point arithmetic rules, see "fimath Rules for Fixed-Point Arithmetic".

This feature also includes several new functions:

- `isfimathlocal` — Use this function to find out whether a `fi` object has its own explicitly attached `fimath` object. Any `fi` object that does not have its own `fimath` object is instead associated with the global fimath.
- `sfi`, `ufi` — Use these constructors to create signed and unsigned `fi` objects that associate themselves with the global fimath. See "New sfi and ufi Constructors for Creating fi Objects" on page 33 for more information.
- `removedefaultfimathpref` — Use this function to remove a user-configured global fimath from your MATLAB preferences. See Working with the Global fimath in the *Fixed-Point Toolbox User's Guide* for more information.

Compatibility Consideration

The new relationship between `fi` objects and the global fimath results in the following incompatibilities:

Construction of fi Objects. To create `fi` objects withFixed-Point Toolbox software, the `fi` constructor must know which `RoundMode` and `OverflowMode` properties to use. You can specify these `fimath` object properties in a `fi` object constructor. In this case, the constructor uses the `RoundMode` and `OverflowMode` properties you specify to create the `fi` object.

The new relationship between `fi` objects and the global `fimath` state introduces a compatibility consideration. This compatibility issue relates to the construction of `fi` objects from a floating-point value when you do not specify any `fimath` object properties in the `fi` object constructor. When you constructed `fi` objects in such a way in R2008b and R2009a, the `fi` constructor used the `RoundMode` and `OverflowMode` properties of the old default `fimath` object.

When you create a `fi` object in this way in R2009b, you may get different results. As of this release, the `fi` constructor creates the `fi` object using a `RoundMode` of `nearest` and an `OverflowMode` of `saturate`. Thus, the setting of the `RoundMode` and `OverflowMode` properties of the global fimath are not used to create `fi` objects from floating-point values.

You can still create a `fi` object from a floating-point value in R2009b with a `RoundMode` and `OverflowMode` other than `nearest` and `saturate`. To do so, specify your desired `RoundMode` and `OverflowMode` properties in the `fi` constructor.

Behavior change for add, mpy and sub Functions. In previous releases, the output of the `fimath` object functions `add`, `mpy` and `sub` was assigned the `fimath` object specified in the function call. For example, the syntax `c = add(F,a,b)` where `F` is a `fimath` object, resulted in a `fi` object `c` being created and assigned the `fimath` object `F`.

In R2009b, the output `fi` object `c` is always associated with the global fimath. It no longer gets assigned the `fimath` object `F`.

Specifying fimath Properties in the Embedded MATLAB Function Block. This release removes the **FIMATH for fixed-point input signals** and **FIMATH for fi and fimath constructors** parameters from the Embedded MATLAB Function Block. You now use the **Embedded MATLAB function block fimath** radio buttons to set the following:

- The `fimath` properties to be associated with fixed-point inputs
- The `fimath` properties to be associated with all `fi` and `fimath` objects constructed in the block

You can choose one of the following options for the **Embedded MATLAB function block fimath** parameter:

- **Same as MATLAB** — When you select this option, the block uses the same `fimath` properties as the current global fimath.
- **Specify other** — When you select this option, you can specify your own set of `fimath` object properties. These properties get associated with all fixed-point block inputs and all `fi` and `fimath` objects constructed in the block.

Run a model that had the **FIMATH for fi and fimath constructors** parameter set to `Same as MATLAB factory default` in R2009b. You may notice different results. This difference occurs because all `fi` and `fimath` objects constructed in the Embedded MATLAB function block now use the **Embedded MATLAB function block fimath**.

To achieve the same results as previous releases, set the **Embedded MATLAB function block fimath** to the MATLAB factory default when you run your model in R2009b. Otherwise, all `fi` and `fimath` objects constructed within your Embedded MATLAB function blocks may be associated with different `fimath` object properties.

To inform you of this change in behavior, the block issues a warning at the MATLAB command line. To turn off the warning, run `slupdate` on your model.

New sfi and ufi Constructors for Creating fi Objects

Fixed-Point Toolbox software now supports two new `fi` object constructor functions:

- `sfi` — Construct signed fixed-point numeric object
- `ufi` — Construct unsigned fixed-point numeric object

These new constructors simplify the construction of `fi` objects. You no longer need to specify the signedness and `fimath` object properties of the `fi` object in the constructor. All `fi` objects you construct with one of these functions are automatically associated with the global fimath. They do not have a `fimath` object of their own. For more information on the global fimath and how it can help you simplify arithmetic with `fi` objects, see:

"New Relationship Between fi Objects and a Global fimath Simplifies Fixed-Point Arithmetic" on page 30

New conv Function for Fixed-Point Inputs

Fixed-Point Toolbox software now provides support for the `conv` function.

Built-in Integer and Floating-Point Support Added for Bitshift Operations

The following Fixed-Point Toolbox functions now support built-in integers:

- `bitsll`
- `bitsrl`
- `reinterpretcast`

The following Fixed-Point Toolbox function now supports both built-in integers and floating-point data types:

- `bitsra`

Embedded MATLAB Support for Variable-Size Arrays and Matrices

The Embedded MATLAB subset now supports variable-size arrays and matrices with known upper bounds. With this feature, you can define inputs, outputs, and local variables in Embedded MATLAB functions to represent data that varies in size at runtime. You can use variable-size data in:

- Embedded MATLAB Function blocks in Simulink
- Embedded MATLAB functions in Stateflow® charts

- Embedded MATLAB compliant M functions, from which you can generate MEX code with `emlmex` and C code with `emlc` (requiresReal-Time Workshop® software)

For more information, see "How Working with Variable-Size Data Is Different for Code Generation" in the *Embedded MATLAB User's Guide*.

New Compilation Report for Embedded MATLAB Function Blocks

The new compilation report provides compile-time type information for the variables and expressions in your Embedded MATLAB functions. This information helps you find the sources of error messages and understand type propagation issues, particularly for fixed-point data types.

Compatibility Consideration

The new compilation report is not supported by the MATLAB internal browser on Sun™ Solaris™ 64-bit platforms. To view the compilation report, you must have your MATLAB Web preferences configured to use an external browser, for example, Mozilla® Firefox®. To learn how to configure your MATLAB Web preferences, see Web Preferences in the MATLAB documentation.

Version 2.4 (R2009a) Fixed-Point Toolbox Software

This table summarizes what's new in Version 2.4 (R2009a):

New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems
Yes Details below	Yes Summary	Bug Reports Includes fixes

New features and changes introduced in this version are:

Support Added for numerictype Objects with Unspecified Sign

A new `Signedness` property has been added to the `numerictype` object. This new `Signedness` property allows you to set the `Signedness` of a `numerictype` object to `Signed`, `Unsigned`, or `Auto`. By setting the `Signedness` property to `Auto`, you can create `numerictype` objects that have an unspecified sign.

Although you can create `numerictype` objects with an unspecified sign (`Signedness: Auto`), all `fi` objects must have a `Signedness` of `Signed` or `Unsigned`. If you use a `numerictype` object with `Signedness: Auto` to construct a `fi` object, the `Signedness` property defaults to `Signed` at `fi` object creation.

The toolbox still supports the `Signed` property of the `numerictype` object. For more information about the `Signedness` property, see "numerictype Object Properties" in the Fixed-Point Toolbox documentation.

New Fixed-Point Menu Options in the MATLAB Editor

A new **Tools > Fixed-Point Toolbox** menu option in the MATLAB Editor allows you to build `fi`, `fimath`, and `numerictype` object constructors in a GUI and insert them directly into your file. For more information, see the following sections:

- "Constructing fi Objects"
- "Constructing fimath Objects"
- "Constructing numerictype Objects"

New mrdivide and rdivide Functions for Fixed-Point Inputs

Fixed-Point Toolbox software now provides the following `fi` object functions for division:

- `mrdivide` — Forward slash (/) or right-matrix division
- `rdivide` — Right-array division (./)

New Compilation Report for emlc and emlmex

A new compilation report is now available for `emlc` and `emlmex`. The new report provides compile-time type information for the variables and expressions in your MATLAB code. This information simplifies finding sources of error messages and aids understanding of type propagation rules, particularly for fixed-point data types. To generate the report, you must specify the `-report` option after the `emlc` or `emlmex` command. To learn more about the compilation report, see the following sections:

- For information about using the report to view fixed-point data types, see Working with Fixed-Point Compilation Reports in the Fixed-Point Toolbox documentation.
- For more information about using the report with `emlmex`, see Working with Compilation Reports in the Embedded MATLAB subset documentation.
- For more information about using the report with `emlc`, see Working with Compilation Reports in the Simulink Coder documentation.

Compatibility Consideration

The following internal and external browsers do not support the new compilation report:

- MATLAB internal browser (on 64-bit UNIX® platforms only)
- MACI internal browser
- Microsoft® Internet Explorer® 6

To view the new compilation report when your internal browser does not support it, you must configure your MATLAB Web preferences to use an external browser, for example, Firefox. To learn how to configure your MATLAB Web preferences, see "Specify the System Browser for UNIX Platforms" in the MATLAB documentation.

New and Updated Demos

The Fixed-Point ATAN2 Calculation demo is new for this release. This new demo uses the CORDIC algorithm and polynomial approximation to perform a fixed-point calculation of the four quadrant inverse tangent. To run this demo, type `fixpt_atan2_demo` at the MATLAB command line.

The Fixed-Point Data Type Override, Min/Max Logging, and Scaling demo has been updated for this release. The updated demo uses the `emlc` and `emlmex` functions to generate code and demonstrates the Embedded MATLAB subset's ability to generate code beyond 32-bits. To run this demo type `fi_datatype_override_demo` at the MATLAB command line.

Complex fi Objects with Nontrivial Slope and Bias Scaling Are No Longer Supported

Fixed-Point Toolbox operations that result in the creation of a complex `fi` object with nontrivial slope and bias scaling are no longer supported. All complex `fi` objects must now have an integer power-of-two slope and a bias of 0 (binary-point scaling).

Compatibility Consideration

In prior releases, Fixed-Point Toolbox software supported complex `fi` objects with nontrivial slope and bias scaling. As of R2009a, all complex `fi` objects

must have an integer power-of-two slope and a bias of 0. Operations that result in the creation of a `fi` object with nontrivial slope and bias scaling will error out.

Version 2.3 (R2008b) Fixed-Point Toolbox Software

This table summarizes what's new in Version 2.3 (R2008b):

New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems
Yes Details below	No	Bug Reports Includes fixes

New features and changes introduced in this version are:

The Default fimath Object is Now User-Configurable

When you create a `fi` object in MATLAB code or the Embedded MATLAB Function block without providing any `fimath` attributes in the constructor call, a default `fimath` object is assigned to the `fi` object you create. Previously, this default `fimath` object could not be user-configured, and the only way to assign a different `fimath` to your `fi` objects was to specify the desired `fimath` object in every constructor call.

In R2008b, the default `fimath` object used in the `fi` and `fimath` constructors can be user-configured. For more information, see Configure the Default fimath Object in the *Fixed-Point Toolbox User's Guide*.

Embedded MATLAB Subset Support for Data up to 128 Bits

The Embedded MATLAB subset now supports fixed-point word lengths up to 128 bits. This includes:

- Acceleration of fixed-point algorithm execution with the `emlmex` function.
- C code production with the `emlc` function.
- Model simulation and C code production with the MATLAB Function block.

round and convergent Rounding Mode Support Added to the Embedded MATLAB Subset

The Embedded MATLAB subset now supports two additional rounding modes:

- `round` — Rounds to the closest representable number. In the case of a tie, the `round` method rounds positive numbers to the closest representable number in the direction of positive infinity, and rounds negative numbers to the closest representable number in the direction of negative infinity.
- `convergent` — Rounds to the closest representable number. In the case of a tie, `convergent` rounds to the nearest even number.

New reinterpretcast Function

Fixed-Point Toolbox software now provides a `reinterpretcast` function to convert fixed-point data types without changing the underlying data.

New sort Function for Fixed-Point Inputs

Fixed-Point Toolbox software now provides support for the MATLAB `sort` function.

Version 2.2.1 (R2008a+) Fixed-Point Toolbox Software

This table summarizes what's new in Version 2.2.1 (R2008a+)

New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems
No	No	Bug Reports Includes fixes

There were no new features or changes in this version.

Version 2.2 (R2008a) Fixed-Point Toolbox Software

This table summarizes what's new in Version 2.2 (R2008a):

New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems
Yes Details below	No	Bug Reports Includes fixes

New features and changes introduced in this version are:

Enhanced Memory Management

The memory management of the `fi` object has been improved, and generating large multidimensional arrays should no longer hang the MATLAB environment.

When performing computations with `fi` objects from the MATLAB command line, the memory usage in bits for the `fi` variable *a* is on the order of

`max(64, a.wordlength)*numberofelements(a)`

The formula computes the number of bits used to store each element of an array, but does not include the overhead for each instance of the object. Similar to built-in `mxArrays`, there is additional overhead for each array. This overhead is used to store information about data type, `fimath` properties, and other settings that do not depend on the size of the array.

If you are concerned with memory usage, try compiling with `emlmex` or `emlc`. The compiled code uses the smallest C integer type that contains the word

length of the variables. For example, if the `fi` variable *a* is 8-bit, then the compiled code will use approximately 8*`numberofelements`(*a*) bits of memory. The current maximum fixed-point word length in the Embedded MATLAB subset is 32 bits.

New Rounding Functions for fi Objects

Fixed-Point Toolbox software now provides the following `fi` object functions:

- `ceil` — Round toward positive infinity
- `convergent` — Round toward nearest integer with ties rounding to nearest even integer
- `fix` — Round toward zero
- `floor` — Round toward negative infinity
- `nearest` — Round toward nearest integer with ties rounding toward positive infinity
- `round` — Round toward nearest integer with ties rounding to nearest integer with greater absolute value

These functions are also supported by the Embedded MATLAB subset.

New Bitwise Operator bitreplicate

Fixed-Point Toolbox software now provides the `bitreplicate` function to replicate and concatenate the bits of a `fi` object.

`bitreplicate` is also supported by the Embedded MATLAB subset.

New Syntax for Bitwise Operator bitconcat

The Fixed-Point Toolbox function `bitconcat` has new syntax.

For more information see the `bitconcat` reference page.

ndgrid Function Support for Fixed-Point Inputs

Fixed-Point Toolbox software now provides support for the `ndgrid` function.

New fi Constructor Syntax

Fixed-Point Toolbox software has added a new syntax for the `fi` constructor. The syntax `a = fi(V, F, T)` is now defined, and is equivalent to the existing syntax `a = fi(V, T, F)`.

See the `fi` function reference page for more information.

Version 2.1.1 (R2007b+) Fixed-Point Toolbox Software

This table summarizes what's new in Version 2.1.1 (R2007b+)

New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems
No	No	Bug Reports Includes fixes

There were no new features or changes in this version.

Version 2.1 (R2007b) Fixed-Point Toolbox Software

This table summarizes what's new in Version 2.1 (R2007b):

New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems
Yes Details below	Yes—Details labeled as **Compatibility Considerations**, below. See also Summary.	Bug Reports Includes fixes

New features and changes introduced in this version are:

Support for Data Type Override

Fixed-Point Toolbox software now supports data type override in Embedded MATLAB subset. This facilitates fixed-point design and enables a single source for fixed- and floating-point code generation.

New Bitwise Operator Functions

Fixed-Point Toolbox software now provides the following functions:

- `bitandreduce` — Bitwise `AND` of consecutive range of bits
- `bitconcat` — Concatenate bits of two `fi` objects
- `bitorreduce` — Bitwise `OR` of consecutive range of bits
- `bitrol` — Bitwise rotate left

- `bitror` — Bitwise rotate right
- `bitsliceget` — Consecutive slice of bits
- `bitsll` — Bit shift left logical
- `bitsra` — Bit shift right arithmetic
- `bitsrl` — Bit shift right logical
- `bitxorreduce` — Bitwise exclusive `OR` of consecutive range of bits
- `getlsb` — Least significant bit
- `getmsb` — Most significant bit

Embedded MATLAB subset also supports these functions.

bitget Function Updated

The `bitget` function now behaves as follows:

- `bitget` returns a `u1,0`.
- `bitget` supports variable indexing. This means that the position of the bit to get can be a variable instead of a constant.
- The input `fi` object and the position of the bit to get can be vectors or scalars.

For more information, see the `bitget` reference page.

Compatibility Consideration

In prior releases, this function returned a `uint8`. The function now returns a `u1,0`. To get a `uint8`, use the `uint8` function on the `bitget` output.

abs Function Supports Complex Inputs

You can now use the `abs` function to compute the absolute value of a complex `fi` object.

For more information, see the `abs` reference page.

divide Function Updated

The `divide` function now obeys the `DataTypeOverride` settings of the `fipref` object.

For more information, see the `divide` reference page.

Compatibility Consideration

In prior releases, this function did not obey the `DataTypeOverride` settings of the `fipref` object. For example, if the input was `fi ScaledDouble`, but the input `numerictype` object was `fi Fixed`, the output was `fi Fixed`. The output is now `fi ScaledDouble`.

fi Constructor Applies Property/Value Pairs After Numeric Value

When you call the `fi` constructor with both a numeric value and one or more property/value pairs that change the numeric value of the `fi` object, the `fi` constructor applies the property/value pairs after it sets the numeric value of the `fi` object.

For more information, see the `fi` reference page.

Compatibility Consideration

In prior releases, the `fi` constructor applied the property/value pairs before it set the numeric value. For example, the following code used to produce a `fi` object with a value of `0`:

```
a = fi(0,1,16,13,'hex','6488')
```

This code now produces a `fi` object with a value of pi.

Version 2.0 (R2007a) Fixed-Point Toolbox Software

This table summarizes what's new in Version 2.0 (R2007a):

New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems
Yes Details below	Yes—Details labeled as **Compatibility Considerations**, below. See also Summary.	Bug Reports Includes fixes

New features and changes introduced in this version are

Fast Execution for Fixed-Point Algorithms in MATLAB

The new Embedded MATLAB MEX functionality converts MATLAB code to C-MEX functions. These C-MEX functions contain Embedded MATLAB subset optimizations for automatically accelerating fixed-point algorithms to compiled C code speed in MATLAB. For more information, refer to Working with Embedded MATLABMEX in the Embedded MATLAB subset documentation.

New fi Syntaxes that Have fimath as an Argument

The following syntaxes have been added to the `fi` object:

- `a = fi(v,F)`
- `a = fi(v,s,F)`
- `a = fi(v,s,w,F)`
- `a = fi(v,s,w,f,F)`
- `a = fi(v,s,w,slope,bias,F)`
- `a = fi(v,s,w,slopeadjustmentfactor,fixedexponent,bias,F)`

where `v` is value, `s` is signedness, `w` is word length, `f` is fraction length, and `F` is a `fimath` object. Refer to "Working with fi Objects" or the `fi` reference page for more information.

Increased Support for Fixed-Point Toolbox Software Features in the Embedded MATLAB Subset

The following Fixed-Point Toolbox software features are now supported by the Embedded MATLAB subset:

- Dot notation for getting the values of `fimath` properties
- `get` function for `fi` and `fimath` objects
- `diag`, `permute`, `tril`, and `triu` functions

For a complete list of the Fixed-Point Toolbox features supported by the Embedded MATLAB subset, refer to "Functions Supported for Code Acceleration and Code Generation from MATLAB".

Embedded MATLAB Subset Enhanced to Support N-Dimensional Arrays and Function Handles

Embedded MATLAB subset now supports N-dimensional arrays and function handles.

`get` Function Must be Declared Extrinsic in Embedded MATLAB Subset

There is a change to how you must use the `get` function in Embedded MATLAB subset to call properties of any object other than `fi` objects.

Compatibility Consideration

To get properties of non-`fi` objects in Embedded MATLAB subset, you must first declare `get` to be an extrinsic function. As of this release, if you do not do so, your code will error. For more information, refer to "Calling MATLAB Functions" in the Embedded MATLAB subset documentation.

Embedded MATLAB Subset Does Not Support & and | Operators

Embedded MATLAB subset no longer supports & and `|` operators in `if` and `while` conditional statements.

Compatibility Consideration

In prior releases, these operators compiled without error, but their short-circuiting behavior was not implemented correctly. Substitute && and `||` operators instead.

New Demo

The "Fixed-Point Lowpass Filtering Using Embedded MATLAB MEX" demo is new in this release. This demo steps you through generating a C-MEX function from MATLAB code, running the generated C-MEX function, and displaying the results.

Version 1.5 (R2006b) Fixed-Point Toolbox Software

This table summarizes what's new in Version 1.5 (R2006b):

New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems
Yes Details below	No	Bug Reports Includes fixes

New features and changes introduced in this version are

Licensing Changes

You now can use `fi` objects with the `DataType` property set to `double` *without* a Fixed-Point Toolbox license when the `fipref` `LoggingMode` property is set to `off`. For details about the Fixed-Point Toolbox licensing model, refer to "Licensing" in the product documentation.

Fixed-Point Square Root Support

In this release, fixed-point square root support has been added to

- Fixed-Point Toolbox software, via the `sqrt` function
- Embedded MATLAB subset, via support for the Fixed-Point Toolbox `sqrt` function
- Simulink, via fixed-point support for the `sqrt` mode of the Math Function block

These products use the same bisection algorithm to implement their fixed-point square root functionality and yield identical results.

Limited Dot Notation Support Added to Fixed-Point Embedded MATLAB Subset

Dot notation is now supported in Embedded MATLAB subset for getting the values of `numerictype` object properties. Dot notation is not supported for `fi` or `fimath` objects, and it is not supported for setting properties.

get Function Support Added to Fixed-Point Embedded MATLAB Subset

The Fixed-Point Toolbox `get` function is now supported for use with Embedded MATLAB subset with the following limitations:

- Only supported for use with `numerictype` objects
- The syntax `structure = get(o)` is not supported

New Default Syntax for fi Object

You can now use the syntax `fi` without any input arguments to return a default `fi` object with no value, 16-bit word length, and 15-bit fraction length.

Version 1.4 (R2006a) Fixed-Point Toolbox Software

This table summarizes what's new in Version 1.4 (R2006a):

New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems
Yes Details below	Yes—Details labeled as **Compatibility Considerations**, below. See also Summary.	Bug Reports Includes fixes

New features and changes introduced in this version are

[Slope Bias] Math Support Added

Arithmetic using the `+`,`-`, `.*`, and `*` operators is now supported for objects with [Slope Bias] scaling. Refer to "[Slope Bias] Arithmetic" in the product documentation for more information.

In support of this feature, the following properties have been added to the `fimath` object:

- `ProductBias` — Bias of the product data type

- `ProductFixedExponent` — Fixed exponent of the product data type
- `ProductSlope` — Slope of the product data type
- `ProductSlopeAdjustmentFactor` — Slope adjustment factor of the product data type
- `SumBias` — Bias of the sum data type
- `SumFixedExponent` — Fixed exponent of the sum data type
- `SumSlope` — Slope of the sum data type
- `SumSlopeAdjustmentFactor` — Slope adjustment factor of the sum data type

Refer to "Property Reference" in the product reference documentation for more information.

Scaled Double Data Type Support Added to the fi Object

The `fi` object now supports the scaled double data type. The value `ScaledDouble` has been added to the `DataType` property of the `numerictype` object. The following values have also been added to the `DataTypeMode` property of the `numerictype` object:

- `Scaled double: binary point scaling`
- `Scaled double: slope and bias scaling`
- `Scaled double: unspecified scaling`

Math operations are supported for `fi` objects with data type `ScaledDouble`.

Global DataTypeOverride Property Added to the fipref Object

The `fipref` object now has the property `DataTypeOverride`, which allows you to override `fi` objects with scaled doubles, singles, or doubles. Refer to "Using fipref Objects to Set Data Type Override Preferences" in the product documentation for more information.

Embedded MATLAB Subset Supports More Fixed-Point Toolbox Functions

The following Fixed-Point Toolbox functions are now supported by Embedded MATLAB subset:

- `bitand`
- `bitcmp`
- `bitget`
- `bitor`
- `bitset`
- `bitshift`
- `bitxor`
- `rescale`

Refer to "Functions Supported for Code Acceleration and Code Generation from MATLAB" in the product documentation for more information.

Embedded MATLAB Subset Does Not Support a CastBeforeSum Value of `'false'`

You can no longer set the `fimath` object property `CastBeforeSum` to `false` or `0` in Embedded MATLAB Function blocks. The reason for this restriction is that Embedded MATLAB subset does not produce the same numerical results as MATLAB when `CastBeforeSum` is `false`.

Compatibility Considerations

In the previous release, `CastBeforeSum` was set to `false` for default `fimath` objects in Embedded MATLAB subset If you have existing models that contain Embedded MATLAB Function blocks in which `CastBeforeSum` is `false`, you will now get an error when you compile or update your model. To correct this issue, you must set `CastBeforeSum` to `true`. To automate this process, you can run the utility `slupdate` either from the Model Advisor or by typing the following command at the MATLAB command line:

```
slupdate ('modelname')
```

where '*modelname*' is the name of the model containing the Embedded MATLAB Function block that generates the error. `slupdate` prompts you to update this property by selecting one of these options:

Option	Action
Yes	Updates the first occurrence of `CastBeforeSum=false` in Embedded MATLAB Function blocks in the model and then prompts you for each subsequent instance found in the model.
No	Does not update any occurrences of `CastBeforeSum=false` in the model.
All	Updates all occurrences of `CastBeforeSum=false` in the model.

Note `slupdate` detects `CastBeforeSum=false` only in *default* `fimath` objects defined for Simulink signals in Embedded MATLAB Function blocks. If you modified the `fimath` object in an Embedded MATLAB Function block, update `CastBeforeSum` manually in your model and fix the errors as they are reported.

`'round'` Value Added to the fimath Object RoundMode Property

The `RoundMode` property value `round` has been added to the `fimath` object. The behavior of this rounding mode is identical to the MATLAB `round` function. For more information refer to "RoundMode" in the product documentation.

numerictype Object Syntax Change

Previously, if you created a `numerictype` object without specifying a value for the `FractionLength` property, the fraction length would be automatically set to `15`. Now however, if you do not set the `FractionLength` property when creating a `numerictype` object, the scaling will remain unspecified. For example:

```
T = numerictype(1, 16)

T =
```

```
          DataTypeMode: Fixed-point: unspecified scaling
                Signed: true
            WordLength: 16

T.scaling

ans =

Unspecified

T.FractionLength

ans =

     0
```

Compatibility Considerations

Any instances of this syntax in your existing code will now return a different result.

Minimums and Maximums Now Logged After Quantization

Previously, the `fi` and `quantizer` objects logged minimums and maximums before quantization. They now log after quantization.

Compatibility Considerations

If your fixed-point data overflows and you want to log minimums and maximums for the full floating-point range, use the `'ScaledDoubles'` or `'TrueDoubles'` values of the `fipref` object `DataTypeOverride` property. For example, the following fixed-point variable overflows. The saturated minimum and maximum values are logged:

```
p = fipref;
p.LoggingMode = 'On';
p.DataTypeOverride = 'ForceOff';
```

```
a = fi(-2:2, true, 16, 15)
Warning: 3 overflows occurred in the fi assignment operation.

a =

          -1           -1            0      0.99997      0.99997

          DataTypeMode: Fixed-point: binary point scaling
                Signed: true
            WordLength: 16
        FractionLength: 15

             RoundMode: nearest
          OverflowMode: saturate
           ProductMode: FullPrecision
  MaxProductWordLength: 128
               SumMode: FullPrecision
      MaxSumWordLength: 128
         CastBeforeSum: true

logreport(a)

      minlog      maxlog   lowerbound   upperbound   noverflows   nunderflows
  a       -1   0.9999695           -1    0.9999695            3             0
```

Now set `DataTypeOverride` to `'ScaledDoubles'`. Note that overflows are reported, but the data is not quantized. The minimum and maximum logs show the full possible range of the data without quantization:

```
p = fipref;
p.LoggingMode = 'On';
p.DataTypeOverride = 'ScaledDoubles';

b = fi(-2:2, true, 16, 15)
Warning: 3 overflows occurred in the fi assignment operation.

b =

    -2    -1     0     1     2
```

```
          DataTypeMode: Scaled double: binary point scaling
                Signed: true
            WordLength: 16
        FractionLength: 15

             RoundMode: nearest
          OverflowMode: saturate
           ProductMode: FullPrecision
  MaxProductWordLength: 128
               SumMode: FullPrecision
      MaxSumWordLength: 128
         CastBeforeSum: true
```

```
logreport(b)
```

```
     minlog   maxlog   lowerbound   upperbound   noverflows   nunderflows
  b      -2        2           -1    0.9999695            3             0
```

For an in-depth example of using logging and data type override to help set appropriate scalings for fixed-point quantities, see the Fixed-Point Toolbox demo “Fixed-Point Data Type Override, Min/Max Logging, and Scaling”.

Version 1.3 (R14SP3) Fixed-Point Toolbox Software

This table summarizes what's new in Version 1.3 (R14SP3):

New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems
Yes Details below	Yes—Details labeled as **Compatibility Considerations**, below. See also Summary.	Bug Reports Includes fixes

New features and changes introduced in this version are

Fixed-Point Toolbox Function Support Added to Embedded MATLAB Subset

The MATLAB Function block lets you compose a MATLAB language function in a Simulink model that generates embeddable code using the Embedded MATLAB subset. When you simulate the model or generate code for a target environment, a function in an Embedded MATLAB Function block generates efficient C code. This code meets the strict memory and data type requirements of embedded target environments. In this way, Embedded MATLAB Function blocks bring the power of MATLAB for the embedded environment into Simulink.

For more information about the Embedded MATLAB Function block and the Embedded MATLAB subset, refer to the following documentation:

- MATLAB Function block reference page in the Simulink documentation
- "Using the MATLAB Function Block" in the Simulink documentation
- "About Code Generation from MATLAB Algorithms" in the Embedded MATLAB subset documentation

You can now use a significant number of Fixed-Point Toolbox functions with Embedded MATLAB subset. Refer to "Functions Supported for Code Acceleration and Code Generation from MATLAB" in the Using Fixed-Point Toolbox documentation.

Note To simulate models using fixed-point data types in Simulink, including when using the Embedded MATLAB Function block, you must have a Simulink Fixed Point™ license.

Double, Single, and Boolean Data Type Support Added to the fi Object

The `fi` object now supports `double`, `single`, and `boolean` data types. The values `double`, `single`, and `boolean` have been added to the `DataType` and `DataTypeMode` properties of the `numerictype` object. Math operations are supported for `fi` objects with data type `single` or `double`, but not `boolean`.

Fixed-Point Doubles Override, Min/Max Logging, and Scaling Demo

Since floating-point data types are now supported in Fixed-Point Toolbox software, it is possible to use doubles override and min/max scaling to help you choose the appropriate scalings for fixed-point variables in your algorithms. This is especially helpful when converting a floating-point algorithm to fixed point. A new demo "Fixed-Point Doubles Override, Min/Max Logging, and Scaling" leads you through an example of this process. You can access this demo from the **Demos** pane of the Help browser under Toolboxes > Fixed-Point.

Helper Functions Added for Accessing Logged Information

In the previous release it became possible to log overflows and underflows as warnings for all assignment, plus, minus, and multiplication operations when the `fipref` `LoggingMode` property is set to `on`. Now when `LoggingMode` is `on`, you can also use the following helper functions to return logged information to you at the MATLAB command line:

- `maxlog` — Returns the maximum real-world value
- `minlog` — Returns the minimum real-world value
- `noperations` — Returns the number of quantized operations
- `noverflows` — Returns the number of overflows
- `nunderflows` — Returns the number of underflows

To clear the log, use the function `resetlog`.

RoundMode Property Value 'round' Now Called 'nearest'

The `RoundMode` property value `round` is now `nearest`. This is a reflection of the fact that this rounding mode is identical to the Simulink rounding mode `round toward nearest`, and different from the behavior of the MATLAB `round` function.

Compatibility Considerations

For this release, any code using the `RoundMode` property value `round` will still work as it did in previous releases. However, you should update each instance of the property value `round` to `nearest` because in a later release, the property value `round` will give a different answer.

Version 1.2 (R14SP2) Fixed-Point Toolbox Software

This table summarizes what's new in Version 1.2 (R14SP2):

New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems
Yes Details below	No	Bug Reports

New features and changes introduced in this version are

Overflow and Underflow Logging

Fixed-Point Toolbox software now allows you to log overflows and underflows as warnings for all assignment, plus, minus, and multiplication operations. Refer to "Using fipref Objects to Set Logging Preferences" in the Fixed-Point Toolbox documentation for more information.

New Functions

The following functions are new in version 1.2 of Fixed-Point Toolbox software:

abs	all	and	any	area
bar	barh	buffer	clabel	comet
comet3	compass	coneplot	contour	contour3
contourc	contourf	diag	end	errorbar
etreeplot	ezcontour	ezcontourf	ezmesh	ezplot
ezplot3	ezpolar	ezsurf	ezsurfc	feather
fplot	gplot	hankel	hist	histc
intmin	ipermute	isnumeric	isobject	line
logical	lowerbound	mesh	meshc	meshz
not	numberofelements	or	patch	pcolor

permute	plot3	plotmatrix	plotyy	polar
pow2	quiver	quiver3	rgbplot	ribbon
rose	scatter	scatter3	sdec	sign
slice	spy	stairs	stem	stem3
streamribbon	streamslice	streamtube	sum	surf
surfc	surfl	surfnorm	text	toeplitz
treeplot	tril	trimesh	triplot	trisurf
triu	uplus	upperbound	voronoi	voronoin
waterfall	xlim	ylim	zlim	

Version 1.1 (R14SP1) Fixed-Point Toolbox Software

This table summarizes what's new in Version 1.1 (R14SP1):

New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems
No	No	Yes Details below

The particularly important bug fixes in this version are

Bitwise Operators Return Correct Answers for [Slope Bias] Signals

In the previous release, bitwise functions such as `bitshift` might have given wrong answers for [Slope Bias] fixed-point signals. This has been corrected in this release.

fi Object Operations with an Empty Array Work Properly

In the previous release, a segmentation violation occurred for any operation with the format

```
a op e
```

where `a` is a `fi` object, `e` is an empty array, and `op` is any operator such as `+`, `-`, `*`, `.*`, `<`, `>`, etc. This has been corrected in this release.

ispropequal Returns Correct Answers for fimath Objects

The `ispropequal` function has been updated to work properly in this release.

Version 1.0 (R14) Fixed-Point Toolbox Software

This table summarizes what's new in Version 1.0 (R14):

New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems
Yes Details below	No	No bug fixes

Fixed-Point Toolbox software provides fixed-point data types in MATLAB and enables algorithm development by providing fixed-point arithmetic. The toolbox enables you to create the following types of objects:

- `fi` — Defines a fixed-point numeric object in the MATLAB workspace. Each `fi` object is composed of value data, a `fimath` object, and a `numerictype` object
- `fimath` — Governs how overloaded arithmetic operators work with `fi` objects
- `fipref` — Defines the display attributes for `fi` objects
- `numerictype` — Defines the data type and scaling attributes of `fi` objects
- quantizer — Quantizes data sets

Features

Fixed-Point Toolbox software provides you with

- The ability to define fixed-point data types, scaling, and rounding and overflow methods in the MATLAB workspace
- Bit-true real and complex simulation
- Basic fixed-point arithmetic with binary point-only signals
  - Arithmetic operators `+`, `-`, `*`, `.*`
  - Division using the `divide` function
- Arbitrary word length up to `intmax('uint16')`
- Relational, logical, and bitwise operators

- Data visualization via the `plot` function
- Statistics functions such as `abs`, `max`, and `min`
- Conversions between binary, hex, double, and built-in integers
- Interoperability with Simulink, Signal Processing Blockset™ software, and Filter Design Toolbox™ software
- Compatibility with the Simulink To Workspace and From Workspace blocks

Getting Help

This section tells you how to get help for Fixed-Point Toolbox software in this document and at the MATLAB command line.

Getting Help in the Fixed-Point Toolbox User's Guide

Fixed-Point Toolbox objects are discussed in the following chapters:

- "Working with fi Objects"
- "Working with fimath Objects"
- "Working with fipref Objects"
- "Working with numerictype Objects"
- "Working with quantizer Objects"

To get in-depth information about the properties of these objects, refer to "Property Reference".

To get in-depth information about the functions of these objects, refer to Function Reference.

Getting Help at the MATLAB Command Line

To get command-line help for Fixed-Point Toolbox objects, type

```
help objectname
```

For example:

```
help fi
```

```
help fimath
help fipref
help numerictype
help quantizer
```

To invoke Help Browser documentation for Fixed-Point Toolbox functions from the MATLAB command line, type

```
doc fixedpoint/functionname
```

For example:

```
doc fixedpoint/int
doc fixedpoint/add
doc fixedpoint/savefipref
doc fixedpoint/quantize
```

Compatibility Summary for Fixed-Point Toolbox Software

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. Details are provided in the description of the new feature or change.

Version (Release)	New Features and Changes with Version Compatibility Impact
Latest Version **V3.5 (R2012a)**	See the **Compatibility Considerations** subheading for this new feature or change: • "Integer Functions Now Return Real-World Values" on page 6 • "Autoscaling Enhancement" on page 7 • "Plus and Minus Operators Enhancements for Code Generation" on page 8 • "fi, fimath, and numerictype Property Changes" on page 8 • "Dynamic Memory Allocation Based on Size for Code Generation" on page 9 • "Conversion of Error and Warning Message Identifiers " on page 9 • "Functions Being Removed" on page 10
V3.4 (R2011b)	See the **Compatibility Considerations** subheading for this new feature or change: • "Conversion of Error and Warning Message Identifiers" on page 15

Version (Release)	New Features and Changes with Version Compatibility Impact
V3.3 (R2011a)	None
V3.2 (R2010b)	See the **Compatibility Considerations** subheading for these new features or changes: • "Changes to the DataTypeMode Property of the numerictype Object" on page 22 • "Functions Being Removed" on page 24
V3.1 (R2010a)	See the **Compatibility Considerations** subheading for this new feature or change: • "Functions Being Removed" on page 29
V3.0 (R2009b)	See the **Compatibility Considerations** subheading for this new feature or change: • "New Relationship Between fi Objects and a Global fimath Simplifies Fixed-Point Arithmetic" on page 30 • "New Compilation Report for Embedded MATLAB Function Blocks" on page 35

Version (Release)	New Features and Changes with Version Compatibility Impact
V2.4 (R2009a)	See the **Compatibility Considerations** subheading for these new features or changes: • "New Compilation Report for emlc and emlmex" on page 37 • "Complex fi Objects with Nontrivial Slope and Bias Scaling Are No Longer Supported" on page 38
V2.3 (R2008b)	None
V2.2 (R2008a)	None
V2.1.1 (R2007b+)	None
V2.1 (R2007b)	See the **Compatibility Considerations** subheading for each of these new features or changes: • "bitget Function Updated" on page 48 • "divide Function Updated" on page 49 • "fi Constructor Applies Property/Value Pairs After Numeric Value" on page 49

Version (Release)	New Features and Changes with Version Compatibility Impact
V2.0 (R2007a)	See the **Compatibility Considerations** subheading for each of these new features or changes: • "`get` Function Must be Declared Extrinsic in Embedded MATLAB Subset" on page 52 • "Embedded MATLAB Subset Does Not Support & and | Operators" on page 52
V1.5 (R2006b)	None
V1.4 (R2006a)	See the **Compatibility Considerations** subheading for each of these new features or changes: • "Embedded MATLAB Subset Does Not Support a CastBeforeSum Value of `'false'`" on page 57 • "numerictype Object Syntax Change" on page 58 • "Minimums and Maximums Now Logged After Quantization" on page 59
V1.3 (R14SP3)	See the **Compatibility Considerations** subheading for this new feature or change: • "RoundMode Property Value 'round' Now Called 'nearest'" on page 64
V1.2 (R14SP2)	None

Version (Release)	New Features and Changes with Version Compatibility Impact
V1.1 (R14SP1)	None
V1.0 (R14)	Not applicable