Simulink® Fixed Point™ Release Notes

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 V7.1 (R2012a)	Yes Details	Yes Summary	Bug Reports Includes fixes
V7.0 (R2011b)	Yes Details	Yes Summary	Bug Reports Includes fixes
V6.5 (R2011a)	Yes Details	Yes Summary	Bug Reports Includes fixes
V6.4 (R2010b)	Yes Details	Yes Summary	Bug Reports Includes fixes
V6.3 (R2010a)	Yes Details	No	Bug Reports Includes fixes
V6.2 (R2009b)	Yes Details	No	Bug Reports Includes fixes
V6.1 (R2009a)	Yes Details	Yes Summary	Bug Reports Includes fixes
V6.0 (R2008b)	Yes Details	Yes Summary	Bug Reports Includes fixes
V5.6.1 (R2008a+)	No	No	Bug Reports Includes fixes
V5.6 (R2008a)	Yes Details	No	Bug Reports Includes fixes
V5.5.1 (R2007b+)	No	No	Bug Reports Includes fixes
V5.5 (R2007b)	Yes Details	Yes Summary	Bug Reports Includes fixes
V5.4.1 (R2007a+)	No	No	Bug Reports Includes fixes

Version (Release)	New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems
V5.4 (R2007a)	Yes Details	No	Bug Reports Includes fixes
V5.3 (R2006b)	Yes Details	No	Bug Reports
V5.2.1 (R2006a+)	No	No	Bug Reports at Web site
V5.2 (R2006a)	Yes Details	No	Bug Reports at Web site
V5.0 (R14)	Yes Details	Yes Summary	Yes Details
V4.1 (R13SP1)	Yes Details	No	Yes Details
V4.0.1 (R13+)	Yes Details	Yes Summary	Fixed bugs
V4.0 (R13)	Yes Details	Yes Summary	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 7.1 (R2012a) Simulink Fixed Point Software

This table summarizes what's new in Version 7.1 (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:

- "Range Derivation at the Subsystem Level" on page 4
- "NumericTypeScope for Determining Fixed-Point Data Types" on page 4
- "Range Analysis Reports Results for Stateflow Data with Local Scope" on page 5
- "Improved Numerical Checking for round, nearest, and convergent Rounding Modes" on page 5
- "Improved Algorithm for Best Precision Scaling" on page 7

Range Derivation at the Subsystem Level

You can now derive range information for individual atomic subsystems and atomic charts. The derived ranges are based on only local design range information specified in the subsystem or chart. This capability enables a more flexible, dynamic modeling style and easier system validation and calibration.

For more information, see "Derive Ranges at the Subsystem Level".

NumericTypeScope for Determining Fixed-Point Data Types

The `NumericTypeScope` allows you to visualize the dynamic range of data in the form of a `log2` histogram. The scope proposes a fixed-point data type based on the dynamic range of the input data and on criteria, such as word length, that you specify.

You can connect the `NumericTypeScope` to signals in your Simulink model to inspect the dynamic range. You can then see whether there are overflows or loss of precision with the current data type.

The scope is useful for:

- Converting a floating-point model to fixed point. Use the scope to help you specify the initial fixed-point data types.
- Simulating a model after fixed-point conversion. Use the scope to inspect the signals to see if the word size is too large or if there is a danger of overflow.

For more information, see `nts`.

Range Analysis Reports Results for Stateflow Data with Local Scope

Previously, the Simulink Fixed Point™ range analysis software took into account Stateflow® data with local scope. The Simulink Fixed Point range analysis software used that data to calculate Stateflow Chart output ranges, but did not report range information for that data. In R2012a, the software does report range information for Stateflow data with local scope.

Improved Numerical Checking for round, nearest, and convergent Rounding Modes

By default, there are now additional checks in the code that is generated for floating-point to fixed-point and floating-point to integer casts that use `round`, `nearest`, or `convergent` rounding modes. These checks guard against incorrectly rounding `0.49999999` to `1` and incorrectly rounding very large, odd numbers to the closest even number.

In previous releases, for a Data Type Conversion block with input `x` and output `y`, output data type `fixdt(1,16,12)`, rounding mode `Nearest`, and **Optimization** configuration parameters **Remove code from floating-point to integer conversions that wraps out-of-range values** and **Remove code from floating-point to integer conversions with saturation that maps NaN to zero** selected, the generated code was:

```
myModel_Y.y = (int16_T)floor(myModel_U.x * 4096.0 + 0.5);
```

In R2012a, the generated code is:

```
real_T u;
real_T v;
u = myModel_U.x * 4096.0;
v = fabs(u);


% Error protection when eps(x)=1
if (v < 4.503599627370496E+15) {
  % Error protection when x=0.4999999
  if (v >= 0.5) {
    myModel_Y.y = (int16_T)floor(u + 0.5);
  } else {
    myModel_Y.y = (int16_T)(u * 0.0);
  }
} else {
  myModel_Y.y = (int16_T)u;
}
```

If you do not require this additional checking, you can use a code replacement library to generate leaner code without these checks.

To generate code without the checks:

1 Open an existing or new `rtwTargetInfo.m` file to be invoked for each Simulink session. The file must reside on the MATLAB path. (For more information about `rtwTargetInfo.m` files, see “Customizing the Simulink User Interface” in the Simulink documentation.)

2 Add code similar to the following to your `rtwTargetInfo.m` file. Modify the `BaseTfl` value to specify the code replacement library currently selected for your model. You can also modify the `Name` value to specify the string that you want to appear in the **Code replacement library** selection menu for your model. When you are finished, save the file.

```
function rtwTargetInfo(cm)
```

```
cm.registerTargetInfo(@loc_register_tfl);

function this = loc_register_tfl

  this(1) = RTW.TflRegistry('RTW');
  this(1).Name = 'ANSI_C_LeanRounding';
  this(1).TableList = {'make_rounding_simplest_tfl'};
  this(1).BaseTfl = 'ANSI_C';
  this(1).TargetHWDevice = {''};

  this(1).Description = 'Make Rounding Lean for Base TFL';
end
end
```

3. Restart Simulink or issue the MATLAB command, `sl_refresh_customizations`.

4. Open your model, go to the **Interface** pane of the Configuration Parameters dialog box, and select the new code replacement library.

For more information, see "Register Code Replacement Libraries" in the Embedded Coder™ documentation.

Improved Algorithm for Best Precision Scaling

In R2012a, using best-precision scaling is less likely to result in proposed data types that could result in overflows. The new algorithm prevents overflows for all rounding modes except `Ceiling`.

For example, consider a Data Type Conversion block with an **Output minimum** of `-128.6`, and rounding mode `Floor`, and data type specified as `fixdt(1,8)`. In previous releases, for an input signal value of `-128.6`, best precision scaling set the output data type to `fixdt(1,8,0)` which resulted in an overflow. In R2012a, for the same model, best precision scaling now prevents such overflows by setting the output data type to `fixdt(1,8,-1)` and the signal value becomes `-128`.

Compatibility Considerations

Best-precision scaling in R2012a calculates a different data type from that calculated in R2011b only if the value being scaled is between `(RepMin - LSB)` and `RepMin`, where `RepMin` is the representable minimum of the proposed data type. Under these conditions, the output data type used by the block might change to avoid overflow. This change might reduce precision and result in the propagation of different data types. It might also affect the data types proposed by the Fixed-Point Advisor and Fixed-Point Tool.

Version 7.0 (R2011b) Simulink Fixed Point Software

This table summarizes what's new in Version 7.0 (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:

Static Range Analysis Provides Derived Minimum and Maximum Values for Signals

The Simulink Fixed Point software can now perform a static range analysis of your model to derive minimum and maximum range values for signals in your model. When you provide design minimum and maximum information for model inputs, outputs, or blocks, the software performs an analysis to derive ranges for signals in the model based on this information using the design parameters specified in your model. You can use the derived range information to gain a better understanding of the operating range of the model, to verify your model design, and to convert your model to fixed point. For more information, see "Range Analysis".

Automatic Data Typing Based on Derived Minimum and Maximum Values

You can use derived minimum and maximum information, in addition to design and simulation minimum and maximum information, to propose data types for your model. The software can propose data types based on any of the following combinations of range data:

- Design minimum and maximum values
- Design and derived minimum and maximum values
- Design and simulation minimum and maximum values
- Design, derived, and simulation minimum and maximum values

With automatic data typing using derived minimum and maximum data instead of simulation data, you do not have to develop and run comprehensive simulation scenarios.

For more information, see "Automatic Data Typing Using Derived Minimum and Maximum Values".

Automatic Word Length Selection

You can now propose minimum word lengths for the data types in your model and its subsystems to avoid overflow for the collected data range information. The proposal uses the default data type that you specify to replace floating-point data types. The Fixed-Point Tool automatically proposes the minimum word lengths while retaining the specified fraction lengths and signedness. The word length selection takes into account the word lengths supported by the target embedded hardware specified in the

Simulink Configuration Parameters dialog box Hardware Implementation pane. For more information, see "Propose Word Lengths".

Fixed-Point Tool Run Management Enhancements

Multiple Runs

The Fixed-Point Tool now supports multiple runs. In previous releases, the tool provided only two runs, Active and Reference. You switched between these runs to compare the results of, for example, a floating-point run and a run using accepted fixed-point settings. In R2011b, you can set up multiple runs. Each run uses one set of model settings to simulate the model or to derive or propose data types. You can easily switch between different run setups using shortcuts.

You can:

- Store multiple runs.
- Specify custom run names.
- Propose data types based on the results in any run.
- Apply data type proposals based on any run.
- Compare the results of any two runs.
- Rename runs directly in the Fixed-Point Tool Contents pane.

For more information, see "Run Management".

Shortcuts to Set Up Runs

You can now use shortcuts to configure the run name as well as model-wide data type override and instrumentation settings prior to simulation.

The Fixed-Point Tool provides:

- Frequently used factory default shortcuts, such as **Model-wide double override and full instrumentation**, which set up your model so that you can override all fixed-point data types with double-precision numbers and log the simulation minimum and maximum values and overflows.

- The ability to add and edit custom shortcuts.

Shortcuts:

- Simplify the workflow. For example, you can collect a floating-point baseline in a clearly named run.
- Provide the ability to configure data type override and instrumentation settings on multiple subsystems in the model hierarchy at the same time. This capability is especially useful for models that have a complicated hierarchy.
- Are a convenient way to store frequently used settings and reuse them. This capability is particularly useful when switching between different settings during debugging.

For more information, see “Shortcuts to set up runs”.

New Column Views

To facilitate viewing results, the Fixed-Point Tool provides the following new column views:

- `Autoscaling With Simulation Min/Max View`
- `Autoscaling With Derived Min/Max View`
- `Data Collection View`
- `Derived Min/Max View`

The Fixed-Point Tool automatically displays these views at the appropriate stage in the workflow. For example, after deriving minimum and maximum values without simulating the model, the Fixed-Point Tool displays the `Derived Min/Max View`. If you then choose to autoscale your model using these derived values, the Fixed-Point Tool displays the `Autoscaling With Derived Min/Max View` after autoscaling.

You can manually switch between views using the **Column View** menu. Also, you can customize views to display only the information that is of interest to you.

Integrated Fixed-Point Conversion Workflow

In R2011b, the Fixed-Point Advisor has been integrated into the Fixed-Point Tool to streamline the float-to-fixed conversion workflow.

- You now open the Fixed-Point Advisor directly from the Fixed-Point Tool and use it to prepare your model for conversion to fixed point. After preparing your model, the Fixed-Point Advisor returns you to the Fixed-Point Tool to complete the conversion to fixed point.
- The Fixed-Point Advisor has been enhanced to support the derived min/max workflow. Use the Fixed-Point Advisor before deriving minimum and maximum values to replace blocks that do not support fixed-point data types and to remove output data type inheritance. You can now use the Fixed-Point Advisor to perform these tasks without simulating the model to create a reference run.
- The Fixed-Point Advisor no longer provides the **Prepare for Code Generation** tasks. The same checks, **Identify blocks that generate expensive saturation and rounding code** and **Identify questionable fixed-point operations**, are available in the Model Advisor or Code Generation Advisor. If required, run these checks after using the Fixed-Point Tool to prepare your model for code generation.
- The Fixed-Point Tool now proposes data types for floating-point signals in a model. You can choose whether to retain these floating-point signals, use the embedded hardware integer type as the default data type for these floating-point signals, or specify a default fixed-point data type for these signals. For more information, see "Default data type for all floating-point signals" in the Simulink reference.

For more information, see "Fixed-Point Advisor Reference" in the Simulink Fixed Point documentation and `fxptdlg` in the Simulink documentation.

Optimization Option to Allow Use of Floating-Point Multiplication to Handle Net Slope Corrections

R2011b introduces a new optimization parameter, **Use floating-point multiplication to handle net slope corrections**. For some processors, use of multiplication to perform net slope correction for floating-point to fixed-point casts improves code efficiency.

For more information, see “Use floating-point multiplication to handle net slope corrections” in the Simulink Graphical User Interface documentation.

Improved Integer and Fixed-Point Saturating Cast

Simulink Coder™ software now eliminates more dead branches in both integer and fixed-point saturation code.

Improved Generated Code for Fixed-Point Multiplication

Simulink Coder software now avoids generating helper functions for fixed-point multiplication when large shift cases make these helper functions unnecessary. For example, in previous releases, if a multiplication has two inputs with data type `uint32` and the specified product output type is `ufix32_E66`, the generated code contained the helper function:

```
Y1 =
   mul_u32_u32_u32_sr64(U1, U2);
```

In R2011b, the generated code contains

```
Y1 = 0U;
```

Code Generated for Mixed Casts No Longer Uses ldexp

The code generated for mixed casts now uses multiplication by a constant that is an exact power of two, instead of using the ANSI C `ldexp` function. Removing `ldexp` from the generated code results in more readable generated code and avoids unnecessary function calls.

In previous releases, the generated code looked like this:

```
tmp = ldexp(cast_P.Constant_Value,2);
```

In R2011b, the generated code is:

```
tmp = cast_P.Constant_Value*4.0;
```

Enhanced Code Generation Optimization

The **Optimize using specified minimum and maximum values** code generation option now takes into account the minimum and maximum values specified for all `Simulink.Parameter` objects, even if the object is part of an expression. For example, consider a Gain block with a gain parameter specified as an expression such as `k1 + 5`. `k1` is a `Simulink.Parameter` object with `k1.min = -10` and `k1.max = 10`. If minimum and maximum values of the parameter specified in the parameter dialog box are `0` and `20`, the range calculated for this parameter expression is `0` to `15`.

For more information, see "Optimize Generated Code Using Specified Minimum and Maximum Values".

Alias Types No Longer Supported with the slDataTypeAndScale Function

The `slDataTypeAndScale` function supports backwards compatibility of Simulink models by mapping a container type and, if needed, fixed-point scaling to a fully specified data type.

Simulink no longer supports calls to `slDataTypeAndScale` when:

- The first argument is a `Simulink.AliasType` object.
- The first argument is a `Simulink.NumericType` object with property `IsAlias` set to true.

Compatibility Considerations

If your model calls the internal function `slDataTypeAndScale`, you might encounter a compilation error for this model even though it previously compiled successfully. In this case, as the error message indicates, update your model to remove the call to `slDataTypeAndScale`.

Conversion of Error and Warning Message Identifiers

For R2011b, error and warning message identifiers have changed in Simulink Fixed Point.

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 `Simulink:fixpoint:Saturationoccurred` identifier has changed to `SimulinkFixedPoint:util:Saturationoccurred`. If your code checks for `Simulink:fixpoint:Saturationoccurred`, you must update it to check for `SimulinkFixedPoint:util:Saturationoccurred` instead. For a mapping of the new identifiers to the original identifiers, see

http://www.mathworks.com/support/solutions/en/data/1-EPUTL0/?solution=1-EPUTL0

.

Version 6.5 (R2011a) Simulink Fixed Point Software

This table summarizes what's new in Version 6.5 (R2011a):

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 View Manager for Fixed-Point Tool" on page 17
- "Improved Fixed-Point Tool User Interface" on page 18
- "Magnitude-Angle to Complex Block Supports CORDIC Algorithm and Fixed-Point Signals" on page 18
- "Enhanced Code Generation Optimization" on page 18
- "Logic for Checking Range of Fixed-Point Data During Simulation Improved" on page 19
- "Target Function Library Support for abs, min, max, and sign functions" on page 19

New View Manager for Fixed-Point Tool

The Fixed-Point Tool now supports customizable views for enhanced control over columns displayed in the Contents pane. Using a defined subset of properties to display streamlines the task of exploring and editing model object properties. The Fixed-Point Tool provides the following standard views that appear automatically at the appropriate point in the fixed-point conversion workflow:

- Simulation View
- Autoscaling View
- Autoscaling with Design Min/Max View

In addition, you can:

- Select the column view based on the task you are performing
- Customize the standard column views
- Create your own column views
- Export and import column views saved in MAT-files, which you can share with other users

For more information, see “Customizing the Contents Pane View” in the `fxptdlg` function reference.

Compatibility Considerations

Column views replace the **Customize Contents** option provided in previous releases.

Improved Fixed-Point Tool User Interface

The updated Fixed-Point Tool user interface helps guide you through a typical floating to fixed-point conversion by grouping actions in the order that you use them. For more information, see “Fixed-Point Tool”.

Magnitude-Angle to Complex Block Supports CORDIC Algorithm and Fixed-Point Signals

The Magnitude-Angle to Complex block now accepts and outputs fixed-point signals when the approximation method is CORDIC.

- For signed fixed-point types, the input angle must fall within the range [–2π, 2π) radians.
- For unsigned fixed-point types, the input angle must fall within the range [0, 2π) radians.

Enhanced Code Generation Optimization

The **Optimize using specified minimum and maximum values** code generation option now takes into account the minimum and maximum values specified for:

- A `Simulink.Parameter` object provided that it is used on its own. It does not use these minimum and maximum values if the object is part of an expression. For example, if a Gain block has a gain parameter specified as `K1`, where `K1` is defined as a `Simulink.Parameter` object in the base workspace, the optimization takes the minimum and maximum values of `K1` into account. However, if the Gain block has a gain parameter of `K1+5` or `K1+K2+K3`, where `K2` and `K3` are also `Simulink.Parameter` objects, the optimization does not use the minimum and maximum values of `K1`, `K2` or `K3`.
- All design ranges specified on block outputs in a conditionally-executed subsystem, except for the block outputs that are directly connected to an Outport block.

For more information, see "Optimize Generated Code Using Specified Minimum and Maximum Values".

Logic for Checking Range of Fixed-Point Data During Simulation Improved

The logic that Simulink uses to determine whether fixed-point values you specify for block parameters are within the valid range for the fixed-point data type is now consistent with the logic used to calculate best-precision scaling.

For more information about block parameter range checking, see "Checking Parameter Values" in the Simulink documentation.

Target Function Library Support for abs, min, max, and sign functions

In R2011a, Embedded Coder software now supports target function library customization control for fixed-point `abs`, `min`, `max`, and `sign` functions.

For more information, see "Register Code Replacement Libraries" in the Embedded Coder documentation.

Version 6.4 (R2010b) Simulink Fixed Point Software

This table summarizes what's new in Version 6.4 (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

- "New Code Generation Optimization Uses Specified Minimum and Maximum Values" on page 21
- "New Data Type Override Capability to Selectively Override Floating-Point or Fixed-Point Types" on page 21
- "New Option to Turn Off Data Type Override for Individual Data Types" on page 21
- "Enhanced Fixed-Point Advisor Support for Discrete Filter, Transfer Fcn, and Other Floating-Point Inheritance Blocks" on page 22
- "Improved Code Generation Efficiency for Saturating Data Type Conversions" on page 24
- "Saturation Block Supports Logging of Minimum and Maximum Values for the Fixed-Point Tool" on page 24
- "Updated Names for Fixed-Point Tool Options" on page 25
- "Enhancements to Simulink.NumericType Class" on page 25
- "Data Type Override No Longer Applies to Booleans" on page 25
- "Simulink.Signal and Simulink.Parameter Objects Now Obey Model Data Type Override Settings" on page 26
- "Fixed-Point Advisor Task Renamed" on page 26
- "New Demos" on page 27

New Code Generation Optimization Uses Specified Minimum and Maximum Values

In R2010b, the Simulink Fixed Point software provides a new code generation optimization that uses specified minimums and maximums to eliminate dead code and unnecessary mathematical operations. To enable this optimization, select the new **Optimize using specified minimum and maximum values** parameter on the Optimization pane of the Configuration Parameters dialog box.

Note This optimization is for ERT-based targets only and requires a Real-Time Workshop® Embedded Coder™ license.

A `demo` shows how this optimization works.

For more information, see "Optimize Generated Code Using Specified Minimum and Maximum Values".

New Data Type Override Capability to Selectively Override Floating-Point or Fixed-Point Types

In R2010b, the Fixed-Point Tool provides a new **Data type override applies to** option. You can use this option to selectively apply data type override settings to all numeric, floating-point or fixed-point data types. Previously, data type override settings always applied to all data types. This ability provides greater control over data type override settings and helps you pinpoint the cause of fixed-point issues.

For more information, see `fxptdlg` in the Simulink documentation.

New Option to Turn Off Data Type Override for Individual Data Types

You can now turn off data type override for individual data types using one of the following methods:

- Setting the new `DataTypeOverride` property on `Simulink.NumericType` or `embedded.numerictype` objects to `Off`.

- Setting the new **Data type override** parameter on the **Data Type Assistant** to `Off`.
- Using the `fixdt` function and setting the `DataTypeOverride` property to `Off`. For example, `fixdt(1,16,2, 'DataTypeOverride', 'Off')`.

You can specify one of the following options when using numeric types in your Simulink model:

- `Inherit` — When you select this option, the data type inherits the data type override setting from its context, that is, from the block, signal or Stateflow chart in Simulink that is using the data type. This is the default setting of this parameter.
- `Off` — When you select this option, the specified data type is used regardless of the data type override setting. The data type ignores the data type override setting of its context.

The ability to turn off data type override for an individual data type provides greater control over the data types in your model when you apply data type override. For example, you can use this option to ensure that data types meet the requirements of downstream blocks.

For more information, see Simulink.NumericType, `fixdt`, and "Using the Data Type Assistant" in the Simulink documentation.

Enhanced Fixed-Point Advisor Support for Discrete Filter, Transfer Fcn, and Other Floating-Point Inheritance Blocks

The Fixed-Point Advisor can now convert models that contain floating-point inheritance blocks, such as Discrete Filter, Transfer Fcn, and other Simulink, Communications Blockset™, Signal Processing Blockset™, and Video and Image Processing Blockset™ blocks.

What is a floating-point inheritance block?

For floating-point inheritance blocks, when inputs are floating point, all internal and output data types are floating point.

The Fixed-Point Advisor can now remove output inheritance and propose data types for floating-point inheritance blocks.

For more information, see “Remove output data type inheritance” in the “Fixed-Point Advisor Reference”.

Improved Code Generation Efficiency for Saturating Data Type Conversions

In R2010b, the Simulink Fixed Point software has improved generated code for fixed-point casts to generate smaller, more efficient cast expressions.

Note This enhancement does not apply to fixed-point casts involving shifts to the left.

The following example demonstrates the difference between the generated code with and without this enhancement. The code generated with the enhancement uses fewer temporary variables, and avoids unnecessary implicit downcasts and upcasts.

Code Generated Without Enhancement	Code Generated With Enhancement
`int32_T tmp;`	`int32_T tmp;`
`int16_T tmp_0;`	
	`tmp = U1 + U2;`
`tmp = U1 + U2;`	`  if (tmp > 32767) {`
`  if (tmp > 32767) {`	`    tmp = 32767;`
`    tmp_0 = MAX_int16_T;`	`  } else {`
`  } else if (tmp <= -32768) {`	`    if (tmp <= -32768) {`
`    tmp_0 = MIN_int16_T;`	`      tmp = -32768;`
`  } else {`	`    }`
`    tmp_0 = (int16_T)tmp;`	`  }`
`  }`	
	`  Y = U3 * tmp;`
`  Y = U3 * tmp_0;`	

Saturation Block Supports Logging of Minimum and Maximum Values for the Fixed-Point Tool

When you set **Fixed-point instrumentation mode** to `Minimums, maximums and overflows` in the Fixed-Point Tool, the Saturation block logs minimum and maximum values. Previously, this block did not support logging of minimum or maximum values.

Updated Names for Fixed-Point Tool Options

The names of the following Fixed-Point Tool options have been updated:

Fixed-Point Tool Parameter	Old Option Name	New Option Name
Data type override	Force off	Off
	Scaled doubles	Scaled double
	True doubles	Double
	True singles	Single

For more information, see `fxptdlg` in the Simulink documentation.

Enhancements to Simulink.NumericType Class

The Simulink.NumericType class now has the following methods:

- isboolean
- isdouble
- isfixed
- isfloat
- isscalingbinarypoint
- isscalingslopebias
- isscalingunspecified
- issingle

Data Type Override No Longer Applies to Booleans

In R2010b, because overriding boolean data types does not affect quantization, data type override no longer applies to the `boolean` data type.

In previous releases, data type override did apply to the `boolean` data type, sometimes causing update diagram errors if downstream blocks did not accept the override type.

Compatibility Considerations

You can no longer override the `boolean` data type in your model. If you want to override this type, you must change it to a type that can represent boolean data, such as, `uint8`.

Simulink.Signal and Simulink.Parameter Objects Now Obey Model Data Type Override Settings

`Simulink.Signal` and `Simulink.Parameter` objects now honor model-level data type override settings. This capability allows you to share fixed-point models that use `Simulink.Signal` or `Simulink.Parameter` objects with users who do not have a Simulink Fixed Point license.

To simulate a model without using Simulink Fixed Point, use the Fixed-Point Tool to set the model-level **Data type override** setting to `Double` or `Single` and the **Data type override applies to** parameter to `All numeric types`.

If you use `fi` objects or embedded numeric data types in your model, set the `fipref` `DataTypeOverride` property to `TrueDoubles` or `TrueSingles` and the `DataTypeOverrideAppliesTo` property to `All numeric types` to match the model-level settings.

For more information, see `fxptdlg` in the Simulink documentation.

Fixed-Point Advisor Task Renamed

The **Summarize blocks with locked scaling** task has been renamed **Review locked data type settings**, and is now the first task in the **Prepare for Data Typing and Scaling** folder.

For more information, see “Review locked data type settings” in the “Fixed-Point Advisor Reference”.

New Demos

The following demo has been added:

Demo...	Shows How You Can ...
`Fixed-Point Optimizations Using Specified Minimum and Maximum Values`	Optimize fixed-point operations in the generated code using the minimum and maximum values specified in a model.

Version 6.3 (R2010a) Simulink Fixed Point Software

This table summarizes what's new in Version 6.3 (R2010a):

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

Trigonometric Function Block Supports CORDIC Algorithm and Fixed-Point Signals

The Trigonometric Function block now accepts and outputs fixed-point signals when you select `sin`, `cos`, or `sincos` and the approximation method is CORDIC.

Elimination of Double-Precision Conversions for Algorithms that Mix Single Precision with Integer or Fixed-Point Data

In R2010a, casting from single to fixed-point data types no longer inserts unnecessary intermediate double-precision variables. Removing these intermediate variables:

- Results in more efficient code, particularly for embedded targets.
- Enables single to fixed-point casts for targets without double-precision arithmetic support.

Improved Automatic Scaling Handles Data Type Constraints for Several Simulink Blocks

Autoscaling with the Fixed-Point Tool and Fixed-Point Advisor now handles data type constraints for ports on several Simulink blocks. For example, autoscaling now takes into account that:

- The index port of the Selector and Assignment blocks support only `double`, `single`, and built-in integer data types.
- The input port of the Data Type Conversion supports only built-in integer data types when the block is configured to output an enumerated type.
- The index port of the Interpolation Using Prelookup supports only integer data types.

This improved autoscaling reduces data type mismatch errors and enables the Fixed-Point Tool and Fixed-Point Advisor to provide additional diagnostic information when you accept autoscaling proposals. For more information, see "Constrained Data Type Summary" in the *Simulink Fixed Point User's Guide*.

Direct Lookup Table (n-D) Block Supports Fixed-Point Signals

The Direct Lookup Table (n-D) block accepts fixed-point data types for the table input port.

Increased Efficiency of Division by Constant Power of 2

In R2010a, the Real-Time Workshop® software no longer unconditionally replaces divisions by constant power of 2 with casts. The software now replaces division by constant power of 2 with a cast only if this replacement results in less generated code. This enhancement relies on the target compiler to optimize the division appropriate to the target processor.

The decision whether to replace the division is based on these guidelines:

- If the replacement by a cast results in extra rounding code, Real-Time Workshop® does not replace the division.
- If the division requires a helper function, Real-Time Workshop® replaces the division with a cast even if the cast requires extra rounding code.

Fixed-Point Advisor Supports Restore Points

The Fixed-Point Advisor now supports restore points. Restore points provide you with the ability to:

- Save a snapshot of your model at any time during conversion from floating point to fixed point.
- Revert any changes made in response to advice from the Fixed-Point Advisor.
- Load and rerun from any restore point without the need to run through the entire conversion process.

For more information, see “Restore Points” in the *Simulink Fixed Point User’s Guide*.

Improved Fixed-Point Advisor Handling of Unsupported Blocks

The Fixed-Point Advisor now provides:

- A wired-subsystem replacement for the State-Space block.
- A preview of the replacement options for an unsupported block, when available.
- A new context menu option to replace an unsupported block.

For more information, see "Address unsupported blocks" in the Fixed-Point Advisor Reference.

Enhanced Target Function Library Support

Target Function Library enhancements include:

- Ability to create custom Target Function Library entries

  TFLs now support custom entries that allow you to specify near-arbitrary match criteria. You first create your own TFL entry class, derived from either `RTW.TflCFunctionEntryML` (for function replacement) or `RTW.TflCOperationEntryML` (for operation replacement). In your derived class, you implement a `do_match` method with a fixed preset signature and customize the match criteria. You also can modify the implementation signature to meet your application needs. For more information, see "Refine CRL Matching and Replacement Using Custom CRL Table Entries" in the Real-Time Workshop® Embedded Coder™ documentation.

- Additional scalar operator replacements
  - New TFL support for replacing scalar complex operations, including addition, subtraction, multiplication, cast, and complex conjugate. Mixed types are supported.
  - Additionally, you can now replace fixed-point shift right for all fixed-point input types.

Stateflow Support for Chart-Level Data with Fixed-Point Word Lengths Up to 128 Bits

Stateflow chart-level data now support up to 128 bits of fixed-point precision for the following scopes:

- Input
- Output
- Parameter
- Data Store Memory

This increase in maximum precision from 32 to 128 bits provides these enhancements:

- Supports generating efficient code for targets with non-standard word sizes
- Allows charts to work with large fixed-point signals

You can explicitly pass chart-level data with these fixed-point word lengths as inputs and outputs of the following functions:

- Embedded MATLAB® functions
- Simulink functions
- Truth table functions that use Embedded MATLAB action language

For more information, see "Using Fixed-Point Data in Stateflow Charts" in the *Stateflow and Simulink Coder User's Guide*.

Root Inport Support for Fixed-Point Data Contained in a Structure

You can now use a root (top-level) Inport block to supply fixed-point data that is contained in a structure. In releases before R2010a, you had to use a `Simulink.Timeseries` object instead of a structure.

To File and From File Blocks Support Fixed-Point Data

The To File and From File blocks now support fixed-point data with a word length of up to 32 bits.

Version 6.2 (R2009b) Simulink Fixed Point Software

This table summarizes what's new in Version 6.2 (R2009b):

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

- "Discrete Transfer Function Block Supports Fixed-Point Intrinsically" on page 35
- "New PID Controller Blocks Support Fixed-Point Intrinsically" on page 35
- "Rapid Accelerator Mode Now Supports All Fixed-Point Word Lengths for Parameters" on page 35
- "Lookup Table (n-D) Block Supports Parameter Data Types Different from Signal Data Types" on page 35
- "Reduced Memory Use and More Efficient Code for Evenly-Spaced Breakpoints in Prelookup and Lookup Table (n-D) Blocks" on page 36
- "Math Function Block Enhancements for Real-Time Workshop Code Generation" on page 37
- "Improved Fixed-Point Advisor Workflow" on page 37
- "New Optimization Option to Allow Integer Division to Handle Net Slopes that are Reciprocals of Integers" on page 37
- "Enhanced Model Advisor Check Identifies Opportunities to Improve Code Efficiency" on page 37
- "New Diagnostic Controls to Detect Precision Loss in Fixed-Point Constants" on page 38
- "Synchronized Zooming for Fixed-Point Tool Time Series Difference Plot" on page 38
- "Changes in Text and Visibility of Dialog Box Prompts for Easier Use with Fixed-Point Advisor and Fixed-Point Tool" on page 39

- "New and Enhanced Demos" on page 41
- "Function Being Removed in a Future Version" on page 41

Discrete Transfer Function Block Supports Fixed-Point Intrinsically

The Discrete Transfer Fcn block now accepts and outputs fixed-point signals.

New PID Controller Blocks Support Fixed-Point Intrinsically

In discrete-time, the new PID Controller and PID Controller (2 DOF) blocks accept real signals of any numeric data type supported by Simulink software, including fixed-point data types.

Rapid Accelerator Mode Now Supports All Fixed-Point Word Lengths for Parameters

Rapid Accelerator mode now supports fixed-point parameters up to 128 bits. To learn more about fixed-point considerations when accelerating your Simulink models, see "Accelerator and Rapid Accelerator Mode Data Type Considerations" in the *Simulink User's Guide*.

Lookup Table (n-D) Block Supports Parameter Data Types Different from Signal Data Types

The Lookup Table (n-D) block supports breakpoint data types that differ from input data types. This enhancement provides these benefits:

- Lower memory requirement for storing breakpoint data that uses a smaller type than the input signal
- Sharing of prescaled breakpoint data between two Lookup Table (n-D) blocks with different input data types
- Sharing of custom storage breakpoint data in Real-Time Workshop® generated code for blocks with different input data types

The Lookup Table (n-D) block supports table data types that differ from output data types. This enhancement provides these benefits:

- Lower memory requirement for storing table data that uses a smaller type than the output signal
- Sharing of prescaled table data between two Lookup Table (n-D) blocks with different output data types
- Sharing of custom storage table data in Real-Time Workshop® generated code for blocks with different output data types

The Lookup Table (n-D) block also supports separate data type specification for intermediate results. This enhancement enables use of a higher precision for internal computations than for table data or output data.

Reduced Memory Use and More Efficient Code for Evenly-Spaced Breakpoints in Prelookup and Lookup Table (n-D) Blocks

For the Prelookup and Lookup Table (n-D) blocks, Real-Time Workshop® generated code now stores only the first breakpoint, spacing, and number of breakpoints when:

- The breakpoint data is not tunable.
- The index search method is `Evenly spaced points`.

This enhancement reduces memory use and provides faster code execution. Previously, the code stored all breakpoint values in a set, regardless of the tunability or spacing of the breakpoints.

The following enhancements also provide more efficient code for the two blocks:

Block	**Enhancement for Code Efficiency**
Lookup Table (n-D)	Removal of unnecessary bit shifts for calculating the fraction
Prelookup and Lookup Table (n-D)	Use of simple division instead of computationally-expensive function calls for calculating the index and fraction

Math Function Block Enhancements for Real-Time Workshop Code Generation

The Math Function block now supports Real-Time Workshop® code generation for fixed-point data types with fractional slope and nonzero bias for the `magnitude^2`, `square`, and `reciprocal` functions.

Improved Fixed-Point Advisor Workflow

In R2009b, the new Fixed-Point Advisor workflow helps you convert your floating-point Simulink model to a fixed-point model more quickly and efficiently. You can now complete your first iteration through the conversion process without accepting all the recommendations. The new workflow ensures that you convert the entire model before preparing the model for code generation. For more information, see “Converting a Model from Floating- to Fixed-Point Using Simulation Data” and “Fixed-Point Advisor Reference” in the *Simulink Fixed Point User’s Guide*.

New Optimization Option to Allow Integer Division to Handle Net Slopes that are Reciprocals of Integers

R2009b introduces a new optimization parameter, **Use integer division to handle net slopes that are reciprocals of integers**. When a change of fixed-point slope is not a power of two, net slope correction is necessary. Normally, net slope correction uses an integer multiplication followed by shifts. Enabling this new optimization replaces the multiplication and shifts with an integer division under certain simplicity and accuracy conditions. For more information, see “Use integer division to handle net slopes that are reciprocals of integers” in the *Simulink Graphical User Interface*.

Enhanced Model Advisor Check Identifies Opportunities to Improve Code Efficiency

The Model Advisor **Identify questionable fixed-point operations** check can:

- Provide advice on when to use the new **Use integer division to handle net slopes that are reciprocals of integers** optimization parameter

For more information, see "Use integer division to handle net slopes that are reciprocals of integers" in the *Simulink Graphical User Interface*.

- Identify opportunities to improve efficiency of generated code for Lookup Table (n-D) blocks in the following cases:

Breakpoint Spacing	Index Search Method
Uneven	Not `Evenly spaced points`
Even, power of 2	Not `Evenly spaced points`
Even, not power of 2	Not `Evenly spaced points`
	`Evenly spaced points`

For more information, see "Identify questionable fixed-point operations" in the *Simulink Coder Reference*.

New Diagnostic Controls to Detect Precision Loss in Fixed-Point Constants

You can now enable the detection of precision loss in net slope and net bias correction. If you enable these diagnostics, the software alerts you when precision loss occurs. It also provides information about the original fixed-point constant value and the error introduced due to quantization or saturation. For more information, see "Detect underflow", "Detect overflow", and "Detect precision loss" in the *Simulink Graphical User Interface*.

Synchronized Zooming for Fixed-Point Tool Time Series Difference Plot

The Fixed-Point Tool now provides synchronized zooming for the Time Series Difference (A-R) plot. By default, zooming on either the `Active and Reference` plot or the `Difference` plot now zooms both plots. For more information, see "Plot Interface" in the *Simulink Reference*.

Changes in Text and Visibility of Dialog Box Prompts for Easier Use with Fixed-Point Advisor and Fixed-Point Tool

The **Lock output scaling against changes by the autoscaling tool** check box is now **Lock output data type setting against changes by the fixed-point tools**. Previously, this check box was visible only if you entered an expression or a fixed-point data type for the output, such as `fixdt(1,16,0)`. This check box is now visible for any output data type specification. This enhancement enables you to lock the current data type settings on the dialog box against changes that the Fixed-Point Advisor or Fixed-Point Tool chooses.

Which blocks are enhanced?

This enhancement applies to the following blocks:

- Abs
- Constant
- Data Store Memory
- Data Type Conversion
- Difference
- Discrete Derivative
- Discrete-Time Integrator
- Divide
- Dot Product
- Fixed-Point State-Space
- Gain
- Inport
- Lookup Table
- Lookup Table (2-D)
- Lookup Table Dynamic
- Math Function

- MinMax
- Multiport Switch
- Outport
- Prelookup
- Product
- Product of Elements
- Relay
- Repeating Sequence Interpolated
- Repeating Sequence Stair
- Saturation
- Saturation Dynamic
- Signal Specification
- Switch

The **Lock scaling against changes by the autoscaling tool** check box is now **Lock data type settings against changes by the fixed-point tools**. Previously, this check box was visible only if you entered an expression or a fixed-point data type, such as `fixdt(1,16,0)`. This check box is now visible for any data type specification. This enhancement enables you to lock the current data type settings on the dialog box against changes that the Fixed-Point Advisor or Fixed-Point Tool chooses.

Which blocks are enhanced?

This enhancement applies to the following blocks:

- Discrete FIR Filter
- Interpolation Using Prelookup
- Lookup Table (n-D)
- Sum
- Sum of Elements

New and Enhanced Demos

The following demos have been added:

Demo...	Shows How You Can...
`Fault-tolerant Fuel Control System`	Perform a floating-point and a fixed-point simulation of a fuel rate control system designed using Simulink and Stateflow.

Function Being Removed in a Future Version

This function will be removed in a future version of Simulink Fixed Point software.

Function Name	What Happens When You Use This Function?	Compatibility Considerations
`fixpt_instrument_purge`	Still works in R2009b	None

Version 6.1 (R2009a) Simulink Fixed Point Software

This table summarizes what's new in Version 6.1 (R2009a):

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

- "Discrete Filter Block Supports Fixed-Point Data Types" on page 42
- "Prelookup and Interpolation Using Prelookup Blocks Support Parameter Data Types That Are Different from Signal Data Types" on page 43
- "Lookup Table (n-D) and Interpolation Using Prelookup Blocks Perform Efficient Fixed-Point Interpolations" on page 44
- "Autoscaling for Simulink Signal Objects is Supported by Fixed-Point Advisor and Fixed-Point Tool" on page 44
- "Rounding Modes Convergent and Round Added to Multiple Blocks" on page 45
- "Simplest Rounding Mode Added to Multiple Blocks" on page 45
- "Multiword Generated Code Enhancements" on page 45
- "Fixed-Point Tool Provides the Ability to Narrow Down Displayed Results Using Filtering Controls" on page 46
- "MinMax Block Performs More Efficient and Accurate Comparison Operations" on page 46
- "New and Enhanced Demos" on page 46

Discrete Filter Block Supports Fixed-Point Data Types

The Discrete Filter block now offers support for fixed-point and integer data types.

In this release, the following enhancements have been made to the Discrete Filter block:

- Improved numerics and run-time performance of outputs and states by reducing the number of divide operations in the filter to at most one
- Support for vector and matrix inputs
- Support for inputs with mixed complexity
- A new **Initial states** parameter allows you to enter nonzero initial states
- A new **Leading denominator coefficient equals 1** parameter provides a more efficient implementation by eliminating all divides when the leading denominator coefficient is one

Prelookup and Interpolation Using Prelookup Blocks Support Parameter Data Types That Are Different from Signal Data Types

The Prelookup block supports breakpoint data types that differ from input data types. This enhancement provides these benefits:

- Enables lower memory requirement for storing breakpoint data that uses a smaller type than the input signal
- Enables sharing of prescaled breakpoint data between two Prelookup blocks with different input data types
- Enables sharing of custom storage breakpoint data in Real-Time Workshop® generated code for blocks with different input data types

The Interpolation Using Prelookup block supports table data types that differ from output data types. This enhancement provides these benefits:

- Enables lower memory requirement for storing table data that uses a smaller type than the output signal
- Enables sharing of prescaled table data between two Interpolation Using Prelookup blocks with different output data types
- Enables sharing of custom storage table data in Real-Time Workshop® generated code for blocks with different output data types

The Interpolation Using Prelookup block also supports separate data type specification for intermediate results. This enhancement enables use of a different precision for internal computations than for table data or output data.

Lookup Table (n-D) and Interpolation Using Prelookup Blocks Perform Efficient Fixed-Point Interpolations

Whenever possible, Lookup Table (n-D) and Interpolation Using Prelookup blocks use a faster overflow-free subtraction algorithm for fixed-point interpolation. To achieve this efficiency, the blocks use a data type of larger container size to perform the overflow-free subtraction, instead of using control-flow branches as in previous releases. Also, Real-Time Workshop® generated code for fixed-point interpolation is now smaller.

Compatibility Considerations

Due to the change in the overflow-free subtraction algorithm, fixed-point interpolation in Lookup Table (n-D) and Interpolation Using Prelookup blocks might, in a few cases, introduce different rounding results from previous releases. Both simulation and code generation use the new overflow-free algorithm, so they have the same rounding behavior and provide bit-true consistency.

Autoscaling for Simulink Signal Objects is Supported by Fixed-Point Advisor and Fixed-Point Tool

In this release, Fixed-Point Advisor and Fixed-Point Tool can propose new scaling for Simulink signal objects in the base or model workspace. If you accept the proposed scaling, Fixed-Point Advisor or Fixed-Point Tool will apply the new scaling to the Simulink signal objects automatically.

For more information, see “Automatic Data Typing Tools” in the Simulink Fixed Point documentation.

Rounding Modes Convergent and Round Added to Multiple Blocks

Rounding modes `Convergent` and `Round` were added to multiple Simulink, Communications Blockset™, Signal Processing Blockset™, and Video and Image Processing Blockset™ blocks. The introduction of these rounding modes allows numerical agreement with advanced embedded hardware and MATLAB.

For more information, see “Rounding Mode: Convergent” and “Rounding Mode: Round” in the Simulink Fixed Point documentation.

Compatibility Considerations

If you use an earlier version of Simulink to open a model that uses the `Convergent` or `Round` rounding modes, the software automatically changes the rounding mode to `Nearest`.

Simplest Rounding Mode Added to Multiple Blocks

The `Simplest` rounding mode was added to multiple Simulink, Communications Blockset™, Signal Processing Blockset™, and Video and Image Processing Blockset™ blocks. Support for this rounding mode maximizes efficiency for blocks handling mixtures of floating point and fixed point.

For more information, see “Rounding Mode: Simplest” in the Simulink Fixed Point documentation.

Multiword Generated Code Enhancements

More Efficient Reuse of Temporary Variables

A reduction in the number of temporary variables and reorganization of function signatures provide more efficient multiword code. This results in faster execution speeds and reduced memory usage. In addition, the new code compiles faster and is easier to inspect manually.

Support for Real-Time Workshop Embedded Coder Code Control Features

More Real-Time Workshop® Embedded Coder™ code control features now apply to multiword functions. These features provide the ability to customize your code, for example, you can customize the code style.

Fixed-Point Tool Provides the Ability to Narrow Down Displayed Results Using Filtering Controls

In this release, the Fixed-Point Tool provides a results filter in the toolbar which specifies the type of results to display. You can use the filter to focus on the types of results that you are interested in at different stages of the autoscaling workflow. Filter options include:

- All results
- Signal Logging results
- Min/Max results
- Overflows
- Conflicts with proposed data types
- Groups that must share the same data type

For more information, see `fxptdlg` in the *Simulink Reference*.

MinMax Block Performs More Efficient and Accurate Comparison Operations

For multiple inputs with mixed floating-point and fixed-point data types, the MinMax block selects an appropriate data type for performing comparison operations, instead of using the output data type for all comparisons, as in previous releases. This enhancement provides these benefits:

- Faster comparison operations, with fewer fixed-point overflows
- Smaller size of Real-Time Workshop® generated code for the MinMax block

New and Enhanced Demos

The following demo has been added:

Demo...	Shows How You Can...
`Multiword Code Generation`	Use Real-Time Workshop® to convert wide integer and fixed-point operations to multiword C code.

Version 6.0 (R2008b) Simulink Fixed Point Software

This table summarizes what's new in Version 6.0 (R2008b):

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

- "Limit of Bits Increased with Embedded MATLAB Function Block" on page 48
- "Limit of Bits Increased with Accelerated Simulation" on page 49
- "Limit of Bits Increased with Code Generation" on page 49
- "Fixed-Point Advisor Enhanced" on page 49
- "Generated Code Enhancement" on page 49
- "Parameter to Lock Output Scaling Added to Six Simulink Blocks" on page 49
- ""What's This?" Context-Sensitive Help Available for Fixed-Point Tool" on page 50
- "Enhanced Support for Stateflow Charts in the Fixed-Point Tool" on page 51
- "Cell Array No Longer Created When Data Logging Is Enabled in the Fixed-Point Tool" on page 51
- "Name Change for Associated Parameters" on page 52
- "Functions Being Removed in a Future Version" on page 53

Limit of Bits Increased with Embedded MATLAB Function Block

Replacement of Embedded MATLAB Function block limit of 32 bits with standard block limit of 128 bits.

Limit of Bits Increased with Accelerated Simulation

Replacement of accelerated simulation limit of 32 bits with normal simulation limit of 128 bits.

Limit of Bits Increased with Code Generation

Replacement of code-generation limit of 32 bits with simulation limit of 128 bits.

Fixed-Point Advisor Enhanced

In R2008b, the Fixed-Point Advisor is enhanced with:

- Improved usability including more descriptive results and intuitive table formatting.
- Improved analysis such as the ability to regenerate simulation data in **Create simulation reference data**.
- Direct links from the Fixed-Point Advisor to the Fixed-Point Tool for improved analysis after the conversion is complete.
- A system selector that allows you to choose the system level from which to start the Fixed-Point Advisor.

For more information, see "Fixed-Point Advisor" in the Simulink Fixed Point documentation.

Generated Code Enhancement

In R2008b, code generation is enhanced to remove excess saturation logic code, reducing RAM and ROM, improving code efficiency.

Parameter to Lock Output Scaling Added to Six Simulink Blocks

For the following Simulink blocks, the dialog box now displays a parameter to lock scaling of outputs against changes by the autoscaling tool:

- Constant
- Data Store Memory
- Inport
- Outport
- Relay
- Signal Specification

For more information about these blocks, see "Block Reference" in the *Simulink Reference*.

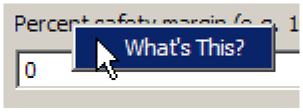
"What's This?" Context-Sensitive Help Available for Fixed-Point Tool

R2008b introduces "What's This?" context-sensitive help for parameters that appear in the Fixed-Point Tool. This feature provides quick access to a detailed description of the parameters, saving you the time it would take to find the information in the Help browser.

To use the "What's This?" help:

1. Place your cursor over the label of a parameter.
2. Right-click. A **What's This?** context menu appears.

   For example, the following figure shows the **What's This?** context menu appearing after right-clicking the **Percent safety margin (e.g. 10 for 10%)** parameter.

3. Click **What's This?** A context-sensitive help window appears showing a description of the parameter.

Enhanced Support for Stateflow Charts in the Fixed-Point Tool

You can now control the signal logging of a Simulink subsystem placed inside a Stateflow chart from the subsystem parent node.

Cell Array No Longer Created When Data Logging Is Enabled in the Fixed-Point Tool

In R2008b, the cell array `FixPtSimRanges` is no longer created automatically in the MATLAB workspace after simulation of a model where data logging is enabled in the Fixed-Point Tool.

Compatibility Considerations

Previously, simulating a model with logging enabled in the Fixed-Point Tool would store maximum values, minimum values, and overflow data in the workspace variable `FixPtSimRanges`. In R2008b, this behavior has changed. However, you can still view this information in one of these ways:

- In the **Contents** pane of the Fixed-Point Tool
- In the MATLAB Command Window by calling the function `showfixptsimranges`

If...	Then...
You do not use scripts to manipulate the data stored in `FixPtSimRanges`	You have no backward incompatibility
You use scripts to manipulate the data stored in `FixPtSimRanges`	You must add a function call to `showfixptsimranges` in your scripts

For example, suppose you have a script as follows:

```
global FixPtSimRanges
outputMaxForBlock1 = FixPtSimRanges{1}.MaxValue;
outputMinForBlock1 = FixPtSimRanges{1}.MinValue;
pathForBlock1 = FixPtSimRanges{1}.Path;
```

In R2008b, you must add a line to the script:

```
global FixPtSimRanges
showfixptsimranges('quiet');
outputMaxForBlock1 = FixPtSimRanges{1}.MaxValue;
outputMinForBlock1 = FixPtSimRanges{1}.MinValue;
pathForBlock1 = FixPtSimRanges{1}.Path;
```

For more information about `showfixptsimranges`, see `showfixptsimranges` in the Simulink Fixed Point Function Reference.

Name Change for Associated Parameters

The Associated parameters in the Autoscale Information dialog box are now Model-Required parameters. Specifically, the **Shared Associated Parameter Initial Value Max** parameter is now **Shared Model-Required Maximum** and **Associated Parameter Initial Value Max** is **Model-Required Maximum**. The names of the minimum values have changed in the same manner. For more information, see "Examine Results to Resolve Conflicts" in the Simulink Fixed Point documentation.

Functions Being Removed in a Future Version

These functions will be removed in a future version of Simulink Fixed Point software.

Function Name	What Happens When You Use This Function?	Compatibility Considerations
`showfixptsimerrors`	Still works in R2008b	Use the Fixed-Point Tool to view overflow data for fixed-point simulations. (See "Fixed-Point Tool" in the *Simulink Fixed Point User's Guide*.)
`showfixptsimranges`	Still works in R2008b	Use the Fixed-Point Tool to view maximum values, minimum values, and overflow data. (See "Fixed-Point Tool" in the *Simulink Fixed Point User's Guide*.)

Version 5.6.1 (R2008a+) Simulink Fixed Point Software

This table summarizes what's new in Version 5.6.1 (R2008a+):

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

Version 5.6 (R2008a) Simulink Fixed Point Software

This table summarizes what's new in Version 5.6 (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 Fixed-Point Tool" on page 55
- "New Fixed-Point Advisor" on page 56
- "Fixed-Point Enhancements in Simulink Blocks" on page 56

Enhanced Fixed-Point Tool

This release introduces the following enhancements to the Fixed-Point Tool:

- In previous releases, you access the Fixed-Point Tool by selecting **Fixed-Point Settings** from the Simulink **Tools** menu. To display the Fixed-Point Tool in this release, from a model's **Tools** menu, select **Fixed-Point > Fixed-Point Tool**. See "Using the Fixed-Point Tool" for more information.
- In this release, the Fixed-Point Tool incorporates design minimum and maximum values in its automatic scaling procedure. Typically, you use a model object's **Output minimum** and **Output maximum** parameters to specify this design range. The tool's **Contents** pane displays these values in new columns titled **DesignMin** and **DesignMax**. See "Propose Data Types" for more information.
- The Fixed-Point Tool provides a new Autoscale Information dialog box. For each model object, this dialog summarizes data type details and explains the tools scaling proposal. See "Examine Results to Resolve Conflicts" for more information.

See "Fixed-Point Tool" in the *Simulink Fixed Point User's Guide* for more information about working with the tool.

New Fixed-Point Advisor

The Fixed-Point Advisor is a new interactive tool you can use to facilitate converting a floating-point model or subsystem to an equivalent fixed-point representation. You can use the Fixed-Point Advisor to prepare a model for conversion and obtain an initial scaling to use as the starting point for refinement and exploration inside the Fixed-Point Tool. For more information, see “Fixed-Point Advisor” in the Simulink Fixed Point documentation.

Fixed-Point Enhancements in Simulink Blocks

This section describes enhancements to fixed-point data type support in Simulink blocks.

Lookup Table (n-D) Block

The Lookup Table (n-D) block now supports fixed-point data types.

Sum Block

The Sum block provides a new parameter for specifying the data type of its accumulator. You can now specify its accumulator data type as any data type that Simulink supports, including fixed-point data types.

Version 5.5.1 (R2007b+) Simulink Fixed Point Software

This table summarizes what's new in Version 5.5.1 (R2007b+):

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

Version 5.5 (R2007b) Simulink Fixed Point Software

This table summarizes what's new in Version 5.5 (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

- "New Signal Logging Options in the Fixed-Point Tool" on page 58
- "Add, Subtract, and Sum Blocks Use an Accumulator Data Type" on page 58
- "Cast Operations with Net Bias Use an Intermediate Data Type" on page 59
- "Non-Matrix-Wise Parameter Scaling Modes Removed" on page 59

New Signal Logging Options in the Fixed-Point Tool

In this release, the Fixed-Point Tool includes new options that provide batch control of signal logging for models and subsystems. These options allow you to enable or disable logging for multiple signals simultaneously, based on the location of signals in a model hierarchy and whether the signals have names. For more information, see the documentation for the `fxptdlg` function in the *Simulink Reference*.

Add, Subtract, and Sum Blocks Use an Accumulator Data Type

In previous releases, the Add, Subtract, and Sum blocks used a user-specified output data type to perform all operations. This behavior might have caused precision loss and saturation during intermediate operations, producing unexpected results. In this release, these blocks use an ideal accumulator data type when performing intermediate operations. Consequently, the Add, Subtract, and Sum blocks now produce more precise results, and the Real-Time Workshop® product generates less saturation code for them.

Cast Operations with Net Bias Use an Intermediate Data Type

Data type conversions with net bias involve an intermediate addition operation. In previous releases, blocks that perform such casts used the output data type's container to compute the addition operation. This behavior might have caused extra saturation, producing unexpected results. In this release, data type conversions can use an ideal data type when performing all intermediate operations. By reducing or eliminating intermediate saturations, cast operations now produce more accurate results, and the Real-Time Workshop® product generates more efficient code for them.

Non-Matrix-Wise Parameter Scaling Modes Removed

In previous releases, the Gain and Weighted Moving Average blocks contained a parameter named **Parameter scaling mode** whose options included the following scaling modes:

- `Best Precision:  Element-wise`
- `Best Precision:  Row-wise`
- `Best Precision:  Column-wise`
- `Best Precision:  Matrix-wise`

In this release, only the `Best Precision:  Matrix-wise` setting is available, and it is now simply named `Best precision`. On the **Parameter Attributes** pane of the Gain or Weighted Moving Average block parameter dialog box, use the **Data Type Assistant** to specify `Best precision` for the **Scaling** control.

Compatibility Considerations

Pre-R2007b models that contain Gain or Weighted Moving Average blocks whose **Parameter scaling mode** option specifies a non-matrix-wise setting will be updated automatically when loaded in R2007b. That is, the Simulink software will change non-matrix-wise settings to `Best precision`, which now represents matrix-wise-best-precision scaling. Consequently, affected blocks might generate results that differ from those obtained in previous releases.

Version 5.4.1 (R2007a+) Simulink Fixed Point Software

This table summarizes what's new in Version 5.4.1 (R2007a+):

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

Version 5.4 (R2007a) Simulink Fixed Point Software

This table summarizes what's new in Version 5.4 (R2007a):

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

Fixed-Point Tool

The Fixed-Point Tool is a new interactive graphical environment for analyzing and scaling fixed-point systems, which replaces the Fixed-Point Settings interface in previous releases. To access the tool, from the Simulink model editor **Tools** menu, select **Fixed-Point Settings**. Alternatively, you use the `fxptdlg` function to access the tool. See "Fixed-Point Tool" in *Simulink Fixed Point User's Guide* for a tutorial that demonstrates how to use the new interface.

Version 5.3 (R2006b) Simulink Fixed Point Software

This table summarizes what's new in Version 5.3 (R2006b):

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

Fixed-Point Enhancements in Simulink Blocks

This section describes enhancements to fixed-point data type support in Simulink blocks.

Math Function Block

The `sqrt` operation in the Math Function block supports fixed-point data types.

New Lookup Table Blocks

The new Prelookup and Interpolation Using Prelookup blocks support fixed-point data types.

Version 5.2.1 (R2006a+) Simulink Fixed Point Software

This table summarizes what's new in Version 5.2.1 (R2006a+):

New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems
No	No	Bug Reports at Web site

Version 5.2 (R2006a) Simulink Fixed Point Software

This table summarizes what's new in Version 5.2 (R2006a):

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

New features and changes introduced in this version are

- "Fixed-Point Block Parameters Supported" on page 64
- "'Simplest' Rounding Mode" on page 64

Fixed-Point Block Parameters Supported

This release allows you to specify fixed-point numbers as the values of Simulink block parameters. In particular, you can specify fixed-point data types in Simulink block parameter dialog boxes and as values of the data type property of `Simulink.Parameter` objects. See "Configuring Blocks with Fixed-Point Parameters" for more information.

'Simplest' Rounding Mode

A new `Simplest` rounding mode is available for the **Round integer calculations toward** parameter of some fixed-point Simulink blocks. This rounding mode attempts to reduce or eliminate the need for extra rounding code in your generated code. The `Simplest` rounding mode is currently available for the following blocks:

- Data Type Conversion
- Product
- Lookup Table
- Lookup Table (2-D)
- Lookup Table Dynamic

For more information, refer to “Rounding Mode: Simplest” in the product documentation.

Version 5.0 (R14) Simulink Fixed Point Software

This table summarizes what's new in Version 5.0 (R14):

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.	Yes Details

New features and changes introduced in this version are

- "Product Restructuring" on page 66
- "Fixed-Point Blocks Fully Integrated into Simulink Software" on page 67
- "API for User-Written Fixed-Point S-Functions" on page 75
- "Fixed-Point Advisor" on page 75
- "Arithmetic with Non-Zero Bias Fully Supported" on page 75
- "Generated Code for Lookup Tables Uses Less ROM" on page 75
- "Functions Moved to Simulink Software" on page 76
- "Obsolete Functions" on page 76
- "Major Bug Fixes" on page 77

Product Restructuring

The Fixed-Point Blockset has been replaced by two new products, Fixed-Point Toolbox™ and Simulink Fixed Point. This product restructuring reflects the broad expansion of fixed-point capabilities in the MATLAB and Simulink software. The Fixed-Point Toolbox product introduces fixed-point operations to the MATLAB language, and the Simulink Fixed Point product enables fixed-point capabilities across much of the Simulink product family.

The Simulink Fixed Point software requires the Fixed-Point Toolbox software. If you are on maintenance, you will automatically receive both of these new products in place of the Fixed-Point Blockset software.

Fixed-Point Blocks Fully Integrated into Simulink Software

All former Fixed-Point Blockset blocks have been moved into the Simulink block libraries with this release. Each of these blocks, as well as all other Simulink blocks, can be used with or without the Simulink Fixed Point software installed. You can share models with any fixed-point and floating-point Simulink blocks among the users in your organization, whether or not they have the Simulink Fixed Point software installed. However, a Simulink Fixed Point software license is required to take full advantage of the fixed-point features of Simulink blocks. For more information, refer to "Sharing Fixed-Point Models" in the *Simulink Fixed Point User's Guide*.

The following table lists all of the blocks in the Fixed-Point Blockset software as of Release 13. It tells you the current name of the block in the Simulink software and the Simulink library in which you can find the block. Most blocks have the same name as in the last release; however, some block names have changed.

Former Fixed-Point Blockset Block	Former Fixed-Point Blockset Library	Simulink Block	Simulink Library
Abs	Math	Abs	Math Operations
Accumulator	Calculus	Discrete-Time Integrator	Discrete
Accumulator Resettable	Calculus	Discrete-Time Integrator	Discrete
Accumulator Resettable Limited	Calculus	Discrete-Time Integrator	Discrete
Add	Math	Add	Math Operations
Bit Clear	Bits	Bit Clear	Logic and Bit Operations
Bit Set	Bits	Bit Set	Logic and Bit Operations
Bitwise Operator	Bits	Bitwise Operator	Logic and Bit Operations

Former Fixed-Point Blockset Block	Former Fixed-Point Blockset Library	Simulink Block	Simulink Library
Compare To Constant	Logic & Comparison	Compare To Constant	Logic and Bit Operations
Compare To Zero	Logic & Comparison	Compare To Zero	Logic and Bit Operations
Constant	Sources	Constant	Sources
Conversion	Data Type	Data Type Conversion	Signal Attributes
Conversion Inherited	Data Type	Data Type Conversion Inherited	Signal Attributes
Cosine	Lookup	Cosine	Lookup Tables
Counter Free	Sources	Counter Free-Running	Sources
Counter Limited	Sources	Counter Limited	Sources
Data Type Duplicate	Data Type	Data Type Duplicate	Signal Attributes
Data Type Propagation	Data Type	Data Type Propagation	Signal Attributes
Dead Zone	Nonlinear	Dead Zone	Discontinuities
Dead Zone Dynamic	Nonlinear	Dead Zone Dynamic	Discontinuities
Decrement Real World	Math	Decrement Real World	Additional Math & Discrete / Additional Math: Increment - Decrement
Decrement Stored Integer	Math	Decrement Stored Integer	Additional Math & Discrete / Additional Math: Increment - Decrement
Decrement Time To Zero	Math	Decrement Time To Zero	Additional Math & Discrete / Additional Math: Increment - Decrement

Former Fixed-Point Blockset Block	Former Fixed-Point Blockset Library	Simulink Block	Simulink Library
Decrement To Zero	Math	Decrement To Zero	Additional Math & Discrete / Additional Math: Increment - Decrement
Derivative	Calculus	Discrete Derivative	Discrete
Detect Change	Edge Detect	Detect Change	Logic and Bit Operations
Detect Decrease	Edge Detect	Detect Decrease	Logic and Bit Operations
Detect Fall Negative	Edge Detect	Detect Fall Negative	Logic and Bit Operations
Detect Fall Nonpositive	Edge Detect	Detect Fall Nonpositive	Logic and Bit Operations
Detect Increase	Edge Detect	Detect Increase	Logic and Bit Operations
Detect Rise Nonnegative	Edge Detect	Detect Rise Nonnegative	Logic and Bit Operations
Detect Rise Positive	Edge Detect	Detect Rise Positive	Logic and Bit Operations
Difference	Calculus	Difference	Discrete
Divide	Math	Divide	Math Operations
Dot Product	Math	Dot Product	Math Operations
Filter Direct Form I	Filters	This block is obsolete.	
Filter Direct Form I Time Varying	Filters	This block is obsolete.	
Filter Direct Form II	Filters	Transfer Fcn Direct Form II	Additional Math & Discrete / Additional Discrete

Former Fixed-Point Blockset Block	Former Fixed-Point Blockset Library	Simulink Block	Simulink Library
Filter Direct Form II Time Varying	Filters	Transfer Fcn Direct Form II Time Varying	Additional Math & Discrete / Additional Discrete
Filter First Order	Filters	Transfer Fcn First Order	Discrete
Filter Lead or Lag	Filters	Transfer Fcn Lead or Lag	Discrete
Filter Real Zero	Filters	Transfer Fcn Real Zero	Discrete
FIR	Filters	Weighted Moving Average	Discrete
Gain	Math	Gain	Math Operations
Gateway In	Data Type	Data Type Conversion	Signal Attributes
Gateway In Inherited	Data Type	Data Type Conversion Inherited	Signal Attributes
Gateway Out	Data Type	Data Type Conversion	Signal Attributes
Increment Real World	Math	Increment Real World	Additional Math & Discrete / Additional Math: Increment - Decrement
Increment Stored Integer	Math	Increment Stored Integer	Additional Math & Discrete / Additional Math: Increment - Decrement
Index Vector	Select	Index Vector	Signal Routing
Integer Delay	Delays & Holds	Integer Delay	Discrete
Integrator Backward	Calculus	Discrete-Time Integrator	Discrete
Integrator Backward Resettable	Calculus	Discrete-Time Integrator	Discrete

Former Fixed-Point Blockset Block	Former Fixed-Point Blockset Library	Simulink Block	Simulink Library
Integrator Backward Resettable Limited	Calculus	Discrete-Time Integrator	Discrete
Integrator Forward	Calculus	Discrete-Time Integrator	Discrete
Integrator Forward Resettable	Calculus	Discrete-Time Integrator	Discrete
Integrator Forward Resettable Limited	Calculus	Discrete-Time Integrator	Discrete
Integrator Trapezoidal	Calculus	Discrete-Time Integrator	Discrete
Integrator Trapezoidal Resettable	Calculus	Discrete-Time Integrator	Discrete
Integrator Trapezoidal Resettable Limited	Calculus	Discrete-Time Integrator	Discrete
Interval Test	Logic & Comparison	Interval Test	Logic and Bit Operations
Interval Test Dynamic	Logic & Comparison	Interval Test Dynamic	Logic and Bit Operations
Logical Operator	Logic & Comparison	Logical Operator	Logic and Bit Operations
Lookup Table	Lookup	Lookup Table	Lookup Tables
Lookup Table Dynamic	Lookup	Lookup Table Dynamic	Lookup Tables
Lookup Table (2-D)	Lookup	Lookup Table (2-D)	Lookup Tables
Matrix Gain	Math	Gain	Math Operations
MinMax	Math	MinMax	Math Operations
MinMax Running Resettable	Math	MinMax Running Resettable	Math Operations

Former Fixed-Point Blockset Block	Former Fixed-Point Blockset Library	Simulink Block	Simulink Library
Multiply	Math	Product	Math Operations
Multiply Matrix	Math	Product	Math Operations
Multi-Port Switch	Select	Multiport Switch	Signal Routing
Product	Math	Product	Math Operations
Product of Elements	Math	Product of Elements	Math Operations
Product of Elements Inverted	Math	Product of Elements	Math Operations
Rate Limiter	Nonlinear	Rate Limiter	Discontinuities
Rate Limiter Dynamic	Nonlinear	Rate Limiter Dynamic	Discontinuities
Relational Operator	Logic & Comparison	Relational Operator	Logic and Bit Operations
Relay	Nonlinear	Relay	Discontinuities
Repeating Sequence Interpolated	Sources	Repeating Sequence Interpolated	Sources
Repeating Sequence Stair	Sources	Repeating Sequence Stair	Sources
Sample Rate Probe	Calculus	Weighted Sample Time	Signal Attributes
Sample Time Add	Calculus	Weighted Sample Time	Signal Attributes
Sample Time Divide	Calculus	Weighted Sample Time	Signal Attributes
Sample Time Multiply	Calculus	Weighted Sample Time	Signal Attributes
Sample Time Probe	Calculus	Weighted Sample Time	Signal Attributes
Sample Time Subtract	Calculus	Weighted Sample Time	Signal Attributes
Saturation	Nonlinear	Saturation	Discontinuities

Former Fixed-Point Blockset Block	Former Fixed-Point Blockset Library	Simulink Block	Simulink Library
Saturation Dynamic	Nonlinear	Saturation Dynamic	Discontinuities
Scaling Strip	Data Type	Data Type Scaling Strip	Signal Attributes
Shift Arithmetic	Bits	Shift Arithmetic	Logic and Bit Operations
Sign	Nonlinear	Sign	Math Operations
Sine	Lookup	Sine	Lookup Tables
State-Space	Filters	Fixed-Point State-Space	Additional Math & Discrete / Additional Discrete
Subtract	Math	Subtract	Math Operations
Sum	Math	Sum	Math Operations
Sum of Elements	Math	Sum of Elements	Math Operations
Sum of Elements Negated	Math	Sum of Elements	Math Operations
Switch	Select	Switch	Signal Routing
Tapped Delay	Delays & Holds	Tapped Delay	Discrete
Unary Minus	Math	Unary Minus	Math Operations
Unit Delay	Delays & Holds	Unit Delay	Discrete
Unit Delay Enabled	Delays & Holds	Unit Delay Enabled	Additional Math & Discrete / Additional Discrete
Unit Delay Enabled External IC	Delays & Holds	Unit Delay Enabled External IC	Additional Math & Discrete / Additional Discrete
Unit Delay Enabled Resettable	Delays & Holds	Unit Delay Enabled Resettable	Additional Math & Discrete / Additional Discrete

Former Fixed-Point Blockset Block	Former Fixed-Point Blockset Library	Simulink Block	Simulink Library
Unit Delay Enabled Resettable External IC	Delays & Holds	Unit Delay Enabled Resettable External IC	Additional Math & Discrete / Additional Discrete
Unit Delay External IC	Delays & Holds	Unit Delay External IC	Additional Math & Discrete / Additional Discrete
Unit Delay Resettable	Delays & Holds	Unit Delay Resettable	Additional Math & Discrete / Additional Discrete
Unit Delay Resettable External IC	Delays & Holds	Unit Delay Resettable External IC	Additional Math & Discrete / Additional Discrete
Unit Delay With Preview Enabled	Delays & Holds	Unit Delay With Preview Enabled	Additional Math & Discrete / Additional Discrete
Unit Delay With Preview Enabled Resettable	Delays & Holds	Unit Delay With Preview Enabled Resettable	Additional Math & Discrete / Additional Discrete
Unit Delay With Preview Enabled Resettable External RV	Delays & Holds	Unit Delay With Preview Enabled Resettable External RV	Additional Math & Discrete / Additional Discrete
Unit Delay With Preview Resettable	Delays & Holds	Unit Delay With Preview Resettable	Additional Math & Discrete / Additional Discrete
Unit Delay Preview Resettable External RV	Delays & Holds	Unit Delay With Preview Resettable External RV	Additional Math & Discrete / Additional Discrete
Wrap To Zero	Nonlinear	Wrap To Zero	Discontinuities
Zero-Order Hold	Delays & Holds	Zero-Order Hold	Discrete

API for User-Written Fixed-Point S-Functions

You can now write your own Simulink C S-functions that directly handle fixed-point data types with a newly published API. For more information, refer to "Writing Fixed-Point S-Functions" in the *Simulink Fixed Point User's Guide*.

Fixed-Point Advisor

The Simulink Fixed Point software now includes Model Advisor checks to help you to configure your fixed-point models to achieve a more efficient design and optimize your generated code. To use the Model Advisor to check your fixed-point models:

1. Select **Model Advisor** from the **Tools** menu of the model you wish to analyze. The Model Advisor appears in the Documents window on the MATLAB desktop.
2. Click **Select All** to enable all Model Advisor checks. For fixed-point code generation, the most important check boxes to select are **Identify questionable fixed-point operations**, **Identify blocks that generate expensive saturation and rounding code**, and **Check the Hardware Implementation**.
3. Click **Check Model**. Any tips for improving the efficiency of your fixed-point model appear in the browser.

Arithmetic with Non-Zero Bias Fully Supported

Code generation has been enhanced to generate bit-true fixed-point code that supports multiplication, division, and reciprocal for signal and parameters with non-zero bias. Previously, these cases lead to code generation errors. Code will now be generated for these cases, and that code will make efficient use of just C integer operations.

Generated Code for Lookup Tables Uses Less ROM

In prior releases, the size of the generated code for models that contained lookup tables with similar attributes was larger than necessary. Such lookup tables produced similar algorithms that appeared throughout the code multiple times. In this release, some common algorithms have been placed into functions which are called by the lookup tables. This enables the same code to be reused multiple times. The overall size of the generated code has been reduced through this enhancement.

Functions Moved to Simulink Software

The following former Fixed-Point Blockset functions are now installed with the Simulink software:

- `fixptbestexp`
- `fixptbestprec`
- `fixpt_interp1`
- `fixpt_look1_func_approx`
- `fixpt_look1_func_plot`
- `fixpt_set_all`
- `float`
- `fxptdlg`
- `num2fixpt`
- `sfix`
- `sfrac`
- `sint`
- `ufix`
- `ufrac`
- `uint`

Obsolete Functions

The functions `fixpt_restore_links` and `fpupdate` are obsolete.

Compatibility Considerations

These functions are no longer needed to update models.

Major Bug Fixes

This section summarizes the major bug fixes introduced in Version 5.0 of the Simulink Fixed Point software.

Simulation Error for 65-Bit+ Multiplication Corrected

In prior releases, fixed-point multiplication could produce the wrong answer under certain simulation conditions. For this error to occur, one input had to have at least 33 bits and the other input at least 32 bits. The correct answer had to be negative, and some additional numerical criteria had to be met. This error could only occur in simulation; it never occurred in generated code. This error has been fully corrected for this release.

Fixed-Point Settings Interface Usable for Large Fonts

In the previous release, the Fixed-Point Settings interface was unusable if your system setup defined large default system fonts. When trying to open the dialog, an error would be reported and the dialog would not appear. The creation of the dialog has now been made robust to large fonts, and this problem is solved.

Lookup Table (2-D) Code Generation Bug Fixed

In a previous release, code generation would error out for the Lookup Table (2-D) block if the input data type had non-zero bias or non-one fractional slope, and the corresponding breakpoints were evenly spaced. This problem has been fixed.

Version 4.1 (R13SP1) Fixed-Point Blockset Software

This table summarizes what's new in Version 4.1 (R13SP1):

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

New features and changes introduced in this version are

- "Improved Treatment of Tunable Parameters" on page 78
- "Generated Code Improved for Lookup Tables and Division" on page 79
- "Major Bug Fixes" on page 79

Improved Treatment of Tunable Parameters

In Release 13, many Simulink and Fixed-Point Blockset blocks were unified. The unified blocks were designed to be fully compatible with models created in earlier releases. However, the unified rules for the treatment of tunable parameters caused compatibility problems for some legacy fixed-point models as discussed in "Backwards Compatibility of Tunable Parameters for Unified Fixed-Point Blocks" on page 81. In this release, these rules have been improved.

A fixed-point model created in Release 12.1 may have experienced problems with tunable parameters when generating code with Versions 5.0 or 5.0.1 of the Real-Time Workshop® software. With the current release, a model created in Release 12.1 will be able to generate code without compatibility problems. Please note that the steps described in "Version 4.0.1 (R13+) Fixed-Point Blockset Software" on page 81 of these Release Notes to solve these compatibility problems do not need to be reversed. The new rules are compatible both with legacy fixed-point models from Release 12.1 and with models that used the work-around described for the previous release.

Generated Code Improved for Lookup Tables and Division

The generated code for utilities that support integer and fixed-point math have been improved to reduce the amount of ROM required. In particular, code that supports lookup tables and division has been improved. The generated code for these operations has been restructured to make greater use of shared functions and less use of inlined code.

Major Bug Fixes

This section summarizes the major bug fixes introduced in Version 4.1 of the Fixed-Point Blockset software.

Plot System Dialog Signal Information Corrected

The **Plot System** dialog is a tool that allows fixed-point simulation results to be easily compared against equivalent floating-point simulation results. Access this dialog by opening the **Fixed-Point Settings** interface from the Simulink **Tools** menu, and then clicking the Show plot dialog icon. For the current model, the dialog provides a list of signals that are logged to the workspace by To Workspace blocks, Scope blocks, and root-level Outport blocks. Signals from this list can be selected, and then plotted in three ways.

There are three plot buttons in the **Plot System** dialog. The Plot Signals button shows the simulation results that are collected using the model's specified data types. The Plot Doubles button shows the simulation results that are collected when the model's specified data types are overridden at the root level by True Doubles or Scaled Doubles. The Plot Both button shows both results simultaneously, making it easy to compare fixed-point behavior against idealized floating-point behavior.

In Release 13, the **Plot System** dialog did not always work properly. Clicking any of the three plot buttons could plot the wrong signals or lead to incorrect error messages. These errors have been corrected. Signals are now associated with the correct plot buttons. In addition, the error messages have been changed to give improved instructions on how to collect the data required by each button.

Fixed-Point Settings Interface Now Usable for Large Fonts

In the previous release, the **Fixed-Point Settings** interface was unusable if your system setup defined large default system fonts. When trying to open the dialog, an error would be reported and the dialog would not appear. The creation of the dialog has now been made robust enough to handle large fonts.

Version 4.0.1 (R13+) Fixed-Point Blockset Software

This table summarizes what's new in Version 4.0.1 (R13+):

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

The major change introduced in this version is

Backwards Compatibility of Tunable Parameters for Unified Fixed-Point Blocks

Unified fixed-point blocks with tunable parameters have compatibility problems under certain conditions in Release 13. The problem arises only if a tunable parameter is mapped to a built-in integer or `single` data type. When tunable parameters are mapped to built-in integers or `single`, the code generated by the Real-Time Workshop® product will be different for unified blocks than it was for Fixed-Point Blockset blocks in prior releases. There are no compatibility problems if a tunable parameter maps to a nonbuilt-in data type, such as a scaled fixed-point integer.

Compatibility Considerations

Tunable parameters are entered in a Simulink model by specifying the name of a MATLAB variable in a block's dialog. This variable can be either a plain MATLAB variable or a Simulink parameter object. In either case, a numerical value will be defined for this tunable parameter by doing an assignment in MATLAB. MATLAB supports several numerical data types including the eight Simulink built-in numerical data types: `double`, `single`, `int8`, `uint8`, `int16`, `uint16`, `int32`, and `uint32`. One of these eight data types can be used when a value is defined for a MATLAB variable. The effect of the data type of the MATLAB variable is significantly different depending on how the tunable parameter is used in the Simulink model.

For Simulink built-in blocks, the legacy rule is to fully respect the data type used for the value of a MATLAB variable. Whatever data type is used in MATLAB when assigning a value to a variable is also used when declaring that parameter in code generated by the Real-Time Workshop® product. The use of that parameter by a block may require the value to be represented using a different data type. If so, additional code is generated to convert the parameter every time it is used by the block. To get the most efficient code for a given block, the value of the MATLAB variable should use the same data type as is needed by the block.

For Fixed-Point Blockset blocks, the legacy rule is to expect no data type information from the MATLAB variable used for the tunable parameter. A fundamental reason for this is that MATLAB does not have native support for fixed-point data types and scaling, so the Simulink built-in legacy rule could not be directly extended to the general fixed-point case. Many fixed-point blocks automatically determine the data type and scaling for parameters based on what leads to the most efficient implementation of a given block. However, certain blocks such as Constant, as well as blocks that use tunable parameters in multiplication, do not imply a unique best choice for the data type and scaling of the parameter. These blocks have provided separate parameters on their dialogs for entering this information.

In Release 13, many Simulink built-in blocks and Fixed-Point Blockset blocks were unified. The Saturation block is an example of a unified block. The Saturation block appears in both the Simulink Library and in the Fixed-Point Blockset Library, but regardless of where it appears it has identical behavior. This identical unified behavior includes the treatment of tunable parameters. The dissimilarity of the legacy rules for tunable parameters has lead to a shortcoming in the unified blocks. Unified blocks obey the Simulink legacy rule sometimes and the Fixed-Point Blockset legacy rule at other times. If the block is using the parameter with built-in Simulink data types, then the Simulink legacy rule applies. If the block is using the parameter with nonbuilt-in data types, such as scaled fixed-point data types, then the Fixed-Point Blockset legacy rule applies. This gives full backwards compatibility with one important exception.

The backwards compatibility issue arises when a model created prior to R13 uses a Fixed-Point Blockset block with a tunable parameter, and the data type used by the block happens to be a built-in data type. If the block is unified, it will now handle the parameter using the Simulink legacy rule rather than the Fixed-Point Blockset legacy rule. This can have a significant impact. For example, suppose the tunable parameter is used in a Saturation block and the data type of the input signal is a built-in `int16`. In prior releases, the Fixed-Point Blockset block would have declared the parameter as an `int16`. For legacy fixed-point models, the MATLAB variables used for tunable parameters invariably gave their value using floating-point `double`. The unified Saturation block would now declare the tunable parameter in the generated code as `double`. This has several negatives. The variable takes up six more bytes of memory as a `double` than as an `int16`. The code for the Saturation block now includes conversions from `double` to `int16` that execute every time the block executes. This increases code size and slows down execution. If the design was intended for use on a fixed-point processor, the use of floating-point variables and floating-point conversion code is likely to be unacceptable. It should be noted that the numerical behavior of the blocks is not changed even though the generated code is different.

For an individual block, the backwards compatibility issue is easily solved. The solution involves understanding that the Simulink legacy rule is being applied. The Simulink legacy rule preserves the data type used when assigning the value to the MATLAB variable. The problem is that an undesired data type will be used in the generated code. To solve this, you should change the way you assign the value of the tunable parameter. Determine what data type is desired in the generated code, then use an explicit type cast when assigning the value in MATLAB. For example, if `int16` is desired in the generated code and the initial value is `3`, then assign the value in MATLAB as `int16(3)`. The generated code will now be as desired.

A preliminary step to solving this issue with tunable parameters is identifying which blocks are affected. In most cases, the treatment of the parameter will involve a downcast from `double` to a smaller data type. On the **Diagnostics** tab of the **Simulation Parameters** dialog is a line item called `Parameter downcast`. Setting this item to `Warning` or `None` will help identify the blocks whose tunable parameters require reassignment of their variables.

In R13, the solution described above did not work for three unified blocks: Switch, Look-Up Table, and Lookup Table (2-D). These blocks caused errors when the value of a tunable parameter was specified using integer data types. This was a false error and has been removed. Using an explicit type cast when assigning a value to the MATLAB variable now solves the issue of generating code with the desired data types.

Version 4.0 (R13) Fixed-Point Blockset Software

This table summarizes what's new in Version 4.0 (R13):

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.	No bug fixes

New features and changes introduced in this version are

- "Installation and Licensing" on page 84
- "Unified Simulink and Fixed-Point Blockset Blocks" on page 85
- "Global Data Type Override and Logging Modes" on page 87
- "Shift Arithmetic Block" on page 88

Installation and Licensing

To support the sharing of models in a large organization, Version 4.0 of the Fixed-Point Blockset product is automatically installed whenever the Simulink software is installed. You can configure models to either take full advantage of all fixed-point features, or to run without a Fixed-Point Blockset software license. Therefore all Simulink software users in your organization can run and work on the same model, regardless of whether they have a Fixed-Point Blockset software license.

You must have a Fixed-Point Blockset software license to run a model if it is configured to log minimums, maximums, or overflows. You control logging with the system-level setting **Logging mode**. If you turn logging off at the top-level system in a model, then no data is logged for any block in any subsystem of the model, and a Fixed-Point Blockset software license is not required. You also need a Fixed-Point Blockset software license to run a model that uses any nonbuilt-in, fixed-point data types. However, you can use the system-level setting **Data type override** to force blocks to use `doubles` or `singles` instead of fixed-point data types. Therefore, by turning the **Data type override** parameter on and the **Logging mode** parameter off at the top level of a model, a Simulink software user without a Fixed-Point Blockset software license can run a model with fixed-point enabled blocks. See "Global Data Type Override and Logging Modes" on page 87 for more information on these settings.

If you have a Fixed-Point Blockset software license, you can run bit-true simulations with your models that contain fixed-point enabled blocks. If a Fixed-Point Blockset software license is not available or desired, you can turn logging off and data type override on at the top level of your model and perform idealized floating point-based simulations.

If you have both a Fixed-Point Blockset software license and a Real-Time Workshop® software license, you can generate bit-true integer code from your models with fixed-point enabled blocks. If you do not have a Fixed-Point Blockset software license but you do have a Real-Time Workshop® software license, you can generate idealized floating-point code from your models with fixed-point enabled blocks.

Unified Simulink and Fixed-Point Blockset Blocks

Many core Simulink and Fixed-Point Blockset blocks with similar functions have been unified in this release. For example, the Sum block in the Simulink Math Operations library and the Sum block in the Fixed-Point Blockset Math library are now the same block. All the functionality from each original block has been maintained in unifying these blocks. Compatibility with fixed-point data types and/or specific fixed-point features are now available with all of these blocks, whether the blocks used are from the Simulink library or the Fixed-Point Blockset library. You do not need to make any changes to your earlier models as a result of this improvement. You can now use any of the unified blocks with either built-in data types or fixed-point data types, which eliminates the need to replace blocks in your models when you want to use different data types. This change does not require Simulink software users to have a Fixed-Point Blockset software license. Refer to "Installation and Licensing" on page 84 for more information.

Fixed-Point Blockset blocks that have been unified no longer have an "F" on their block icon. However, not all Fixed-Point Blockset blocks that have counterparts in the Simulink library have been unified. You can still use the `fixpt_convert` function to replace nonunified Simulink blocks with their corresponding Fixed-Point Blockset blocks in your models.

Nonunified Fixed-Point Blockset blocks have an advantage over their Simulink library counterparts in that they can handle more data types. As discussed above, you can easily switch them between fixed-point data types and singles or doubles using the global data type override setting. However, you may still want to use the Simulink library counterparts of nonunified Fixed-Point Blockset blocks in some cases, because they support faster simulation times for the data types they handle.

The following table lists the unified blocks in this release, and the Simulink and Fixed-Point Blockset libraries in which they are found.

Block	**Simulink Library**	**Fixed-Point Blockset Library**
Abs	Math Operations	Math
Constant	Sources	Sources
Data Store Memory	Signal Routing	N/A
Data Store Read	Signal Routing	N/A
Data Store Write	Signal Routing	N/A
Gain	Math Operations	Math
Inport	Ports & Subsystems, Sources	N/A
Logical Operator	Math Operations	Logic & Comparison
Look-Up Table	Look-Up Tables	LookUp
Look-Up Table (2-D)	Look-Up Tables	LookUp
Manual Switch	Signal Routing	N/A
Memory	Discrete	N/A
Merge	Signal Routing	N/A
Multi-Port Switch	Signal Routing	Select
Outport	Ports & Subsystems, Sinks	N/A
Product	Math Operations	Math
Rate Transition	Signal Attributes	N/A
Relational Operator	Math Operations	Logic & Comparison

Block	Simulink Library	Fixed-Point Blockset Library
Relay	Discontinuities	Nonlinear
Saturation	Discontinuities	Nonlinear
Sign	Math Operations	Nonlinear
Signal Specification	Signal Attributes	N/A
Slider Gain	Math Operations	N/A
Sum	Math Operations	Math
Switch	Signal Routing	Select
Unit Delay	Discrete	Delays & Holds
Zero-Order Hold	Discrete	Delays & Holds

Compatibility Considerations

Breaking library links to Fixed-Point Blockset blocks will almost certainly produce an error when you attempt to run the model. If broken links exist, you will likely uncover them when upgrading to the latest release of the Fixed-Point Blockset software. The `fixpt_restore_links` command can be used to restore links for Fixed-Point Blockset blocks.

Global Data Type Override and Logging Modes

You can now set data type override and logging modes for systems or subsystems in the Fixed-Point Blockset Interface. The **Override data type(s) with doubles** and **Log minimums and maximums** check boxes have been removed from the mask of every Fixed-Point Blockset block.

Compatibility Considerations

The **Override data type(s) with doubles** and **Log minimums and maximums** check boxes have been removed from the mask of every Fixed-Point Blockset block. You can now set these parameters on the system or subsystem level.

When you upgrade to Version 4.0, all doubles override and logging information is cleared from your models. You can reset these controls in the

Fixed-Point Blockset Interface for any system or subsystem. Access the Fixed-Point Blockset Interface from the Simulink **Tools** menu, or by typing `fxptdlg('modelname')` at the MATLAB command line.

If you have been getting or setting the block parameters `DblOver` or `dolog` in your M-code, you must now use the system parameters `DataTypeOverride` and `MinMaxOverflowLogging`.

Shift Arithmetic Block

The Fixed-Point Blockset software now includes the Shift Arithmetic block in the Bits library. The Shift Arithmetic block shifts the bits or binary point of a signal, or both.

Compatibility Summary for Simulink Fixed Point 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 V7.1 (R2012a)	See the **Compatibility Considerations** subheading for these new features or changes: • "Improved Algorithm for Best Precision Scaling" on page 7
V7.0 (R2011b)	See the **Compatibility Considerations** subheading for these new features or changes: • "Alias Types No Longer Supported with the slDataTypeAndScale Function" on page 15 • "Conversion of Error and Warning Message Identifiers" on page 15
V6.5 (R2011a)	See the **Compatibility Considerations** subheading for these new features or changes: • "New View Manager for Fixed-Point Tool" on page 17
V6.4 (R2010b)	See the **Compatibility Considerations** subheading for these new features or changes: • "Data Type Override No Longer Applies to Booleans" on page 25
V6.3 (R2010a)	None
V6.2.(R2009b)	None

Version (Release)	New Features and Changes with Version Compatibility Impact
V6.1 (R2009a)	See the **Compatibility Considerations** subheading for these new features or changes: • "Lookup Table (n-D) and Interpolation Using Prelookup Blocks Perform Efficient Fixed-Point Interpolations" on page 44 • "Rounding Modes Convergent and Round Added to Multiple Blocks" on page 45
V6.0 (R2008b)	See the **Compatibility Considerations** subheading for these new features or changes: • "Cell Array No Longer Created When Data Logging Is Enabled in the Fixed-Point Tool" on page 51 • "Functions Being Removed in a Future Version" on page 53
V5.6.1 (R2008a+)	None
V5.6 (R2008a)	None
V5.5.1 (R2007b+)	None
V5.5 (R2007b)	See the **Compatibility Considerations** subheading for this new feature or change: • "Non-Matrix-Wise Parameter Scaling Modes Removed" on page 59
V5.4.1 (R2007a+)	None
V5.4 (R2007a)	None
V5.3 (R2006b)	None
V5.2.1 (R2006a+)	None

Version (Release)	New Features and Changes with Version Compatibility Impact
V5.2 (R2006a)	None
V5.0 (R14)	See the **Compatibility Considerations** subheading for this new feature or change: • "Obsolete Functions" on page 76
V4.1 (R13SP1)	None
V4.0.1 (R13+)	See the **Compatibility Considerations** subheading for this new feature or change: • "Backwards Compatibility of Tunable Parameters for Unified Fixed-Point Blocks" on page 81

Version (Release)	New Features and Changes with Version Compatibility Impact
V4.0 (R13)	See the **Compatibility Considerations** subheading for each of these new features or changes: • “Unified Simulink and Fixed-Point Blockset Blocks” on page 85 • “Global Data Type Override and Logging Modes” on page 87