Release Notes for Simulink® Fixed Point™

How to Contact MathWorks

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

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

508-647-7000 (Phone)

508-647-7001 (Fax)

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

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

Release Notes for Simulink® Fixed Point™

Trademarks

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

Patents

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

Contents

R2012b

R2012a

R2011b

R2011a

R2010bSP2

R2010b

R2010a

R2009b

R2009a

R2008b

R2008a+

R2008a

R2007b+

R2007b

R2007a+

R2007a

R2006b

R2006a+

No New Features or Changes

R2012b

Version: 7.2
New Features: Yes
Bug Fixes: Yes

Comparison of fixed-point and double-precision signals using Simulation Data Inspector

In R2012b, the Fixed-Point Tool uses the Simulation Data Inspector to plot signals. Using the Simulation Data Inspector to inspect and compare data after converting your floating-point model to fixed point facilitates tracking numerical error propagation by providing the ability to:

- Plot multiple signals in one or more axes
- Compare a signal in different runs
- Compare all logged signal data from different runs
- Export signal logging results to a MAT-file
- Specify tolerances for signal comparison
- Create a report of the current view and data in the Simulation Data Inspector

For more information, see “Viewing Results With the Simulation Data Inspector”.

Instrumentation of simulations for model reference

Simulink® Fixed Point™ now provides support for fixed-point instrumentation of simulation minimums, maximums, and overflows in model reference blocks. The Fixed-Point Tool displays the full model reference hierarchy to facilitate configuration of instrumentation and data type override settings. When you simulate the top model, the Fixed-Point Tool collects instrumentation data for all referenced models in the model hierarchy. If there are multiple instances of the same referenced model, the Fixed-Point Tool displays the merged minimum, maximum and overflow results in the referenced model node. Collecting instrumentation data for referenced models allows you to identify overflows in the referenced models and therefore facilitates debugging fixed-point conversion issues. It also enables automatic data typing for referenced models based on simulation minimums and maximums.

For more information, see “Logging Simulation Minimum and Maximum Values for Referenced Models”.

Derived minimums and maximums for model reference

Simulink Fixed Point now provides support for range analysis for model reference blocks. The Fixed-Point Tool displays the full model reference hierarchy which allows you to derive minimum and maximum data at both the top model and reference model levels. When you perform range analysis for the top model, the Fixed-Point Tool derives ranges for all referenced models in the model hierarchy. If there are multiple instances of the same referenced model, the Fixed-Point Tool displays the merged derived minimum and maximum results in the referenced model node. This new capability helps identify design range propagation issues and enables automatic data typing for referenced models based on derived minimums and maximums.

For more information, see “Derive Ranges for a Referenced Model”.

Automatic data typing for model reference

You can now automatically select data types for blocks and signals in the model reference hierarchy based on design, simulation, and derived range information in the Fixed-Point Tool. You can choose to propose and apply data types at either the top model or the referenced model level. This capability improves the fixed-point workflow and facilitates debugging fixed-point conversion issues.

For more information, see “Propose Data Types for a Referenced Model”.

Fixed-point instrumentation results for MATLAB Function block

Fixed-point instrumentation results, including simulation minimum and maximum values, are now available for MATLAB Function blocks in the MATLAB Function Block report. For more information, see “Log Simulation Minimum and Maximum Values for a MATLAB Function Block”.

Signal name available in Fixed-Point Tool

You can now view signal names in the Fixed-Point Tool by adding the SignalName column to the column view. This capability facilitates viewing the simulation results for signals of interest. For more information, see “View Signal Names in the Fixed-Point Tool”.

R2012a

Version: 7.1
New Features: Yes
Bug Fixes: Yes

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

Compatibility Considerations: Yes

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.

R2011b

Version: 7.0
New Features: Yes
Bug Fixes: Yes

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

Compatibility Considerations: Yes

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

Compatibility Considerations: Yes

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

.

R2011a

Version: 6.5
New Features: Yes
Bug Fixes: Yes

New View Manager for Fixed-Point Tool

Compatibility Considerations: Yes

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.

R2010bSP2

Version: 6.4.1
New Features: No
Bug Fixes: Yes

R2010b

Version: 6.4
New Features: Yes
Bug Fixes: Yes

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;`
`int16_T tmp_0;`
 	
`tmp = U1 + U2;`
`  if (tmp > 32767) {`
`    tmp_0 = MAX_int16_T;`
`  } else if (tmp <= -32768) {`
`    tmp_0 = MIN_int16_T;`
`  } else {`
`    tmp_0 = (int16_T)tmp;`
`  }`
 	
`  Y = U3 * tmp_0;`
	`int32_T tmp;`
	
	`tmp = U1 + U2;`
	`  if (tmp > 32767) {`
	`    tmp = 32767;`
	`  } else {`
	`    if (tmp <= -32768) {`
	`      tmp = -32768;`
	`    }`
	`  }`
	
	`  Y = U3 * tmp;`

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

Compatibility Considerations: Yes

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.

R2010a

Version: 6.3
New Features: Yes
Bug Fixes: Yes

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 documentation.

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.

R2009b

Version: 6.2
New Features: Yes
Bug Fixes: Yes

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 documentation.

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

Compatibility Considerations: Yes

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

R2009a

Version: 6.1
New Features: Yes
Bug Fixes: Yes

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

Compatibility Considerations: Yes

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

Compatibility Considerations: Yes

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.

R2008b

Version: 6.0
New Features: Yes
Bug Fixes: Yes

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.

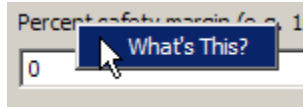
"What's This?" Context-Sensitive Help Available for Fixed-Point Tool

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

Compatibility Considerations: Yes

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

Compatibility Considerations: Yes

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.)

R2008a+

Version: 5.6.1
New Features: No
Bug Fixes: Yes

R2008a

Version: 5.6
New Features: Yes
Bug Fixes: Yes

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.

R2007b+

Version: 5.5.1
New Features: No
Bug Fixes: Yes

R2007b

Version: 5.5
New Features: Yes
Bug Fixes: Yes

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

Compatibility Considerations: Yes

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.

R2007a+

Version: 5.4.1
New Features: No
Bug Fixes: Yes

R2007a

Version: 5.4
New Features: Yes
Bug Fixes: Yes

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.

R2006b

Version: 5.3
New Features: Yes
Bug Fixes: No

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.

R2006a+

Version: 5.2.1
New Features: No
Bug Fixes: No

No New Features or Changes

R2006a

Version: 5.2
New Features: Yes
Bug Fixes: No

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.