Fixed-Point Blockset Release Notes

The "Fixed-Point Blockset 4.0 Release Notes" on page 1-1 describe the changes introduced in the latest version of the Fixed-Point Blockset. The following topics are discussed in these Release Notes:

- "New Features" on page 1-2
- "Upgrading from an Earlier Release" on page 1-6
- "Known Software Problems" on page 1-8

The Fixed-Point Blockset Release Notes also provide information about the earlier versions of the product, in case you are upgrading from a version that was released prior to Release 12.1. If you are upgrading from a release earlier than Release 12.1, you should also see these sections:

- "Fixed-Point Blockset 3.1 Release Notes" on page 2-1
- "Fixed-Point Blockset 3.0 Release Notes" on page 3-1

Printing the Release Notes

If you would like to print the Release Notes, you can link to a PDF version.

Contents

1 Fixed-Point Blockset 4.0 Release Notes

2 Fixed-Point Blockset 3.1 Release Notes

3 Fixed-Point Blockset 3.0 Release Notes

1

Fixed-Point Blockset 4.0 Release Notes

New Features

This section summarizes the new features and enhancements introduced in the Fixed-Point Blockset 4.0.

If you are upgrading from a release earlier than Release 12.1, then you should see “New Features” on page 2-2.

This section is organized into the following subsections:

Installation and Licensing

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

You must have a Fixed-Point Blockset 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 license is not required. You also need a Fixed-Point Blockset 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 user without a Fixed-Point Blockset license can run a model with fixed-point enabled blocks. See “Global Data Type Override and Logging Modes” on page 1-5 for more information on these settings.

If you have a Fixed-Point Blockset license, you can run bit-true simulations with your models that contain fixed-point enabled blocks. If a Fixed-Point Blockset 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 license and a Real-Time Workshop 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 license but you do have a Real-Time Workshop 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 Simulink or from the Fixed-Point Blockset. 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 users to have a Fixed-Point Blockset license. Refer to “Installation and Licensing” on page 1-2 above 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 Simulink libraries have been unified. You can still use the `fixpt_convert` function to replace nonunified Simulink blocks with their Fixed-Point Blockset counterparts in your models.

Nonunified Fixed-Point Blockset blocks have an advantage over their Simulink 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 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
Relay	Discontinuities	Nonlinear
Saturation	Discontinuities	Nonlinear
Sign	Math Operations	Nonlinear

Block	Simulink Library	Fixed-Point Blockset Library
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

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. See “Data Type Override and Logging Parameters” on page 1-6.

Shift Arithmetic Block

The Fixed-Point Blockset 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.

Upgrading from an Earlier Release

This section describes the upgrade issues involved in moving from the Fixed-Point Blockset 3.1 to Version 4.0.

Replacing Obsolete Blocks

If you are using blocks from previous versions of the Fixed-Point Blockset, your model may contain obsolete blocks. The `fpupdate` function can be used to update obsolete blocks from previous Fixed-Point Blockset releases to current Fixed-Point Blockset blocks.

`fpupdate('model')` replaces all obsolete Fixed-Point Blockset blocks contained in the model with current blocks. The model must be opened prior to calling `fpupdate`.

`fpupdate('model',blkprompt)` prompts you for replacement of obsolete blocks. If `blkprompt` is `0` (the default), you will not be prompted. If `blkprompt` is `1`, you will have three options:

- 'y' (default) replaces the block
- 'n' does not replace the block
- 'a' replaces all blocks without further prompting

Restoring Broken Links

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. The `fixpt_restore_links` command can be used to restore links for Fixed-Point Blockset blocks.

Data Type Override and Logging Parameters

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`.

Known Software Problems

You can see a list of known software problems in the Fixed-Point Blockset Version 4.0. If you are viewing these Release Notes in PDF form, please refer to the HTML form of the Release Notes, using either the Help browser or the MathWorks Web site, and use the link provided.

2

Fixed-Point Blockset 3.1 Release Notes

New Features

This section introduces the new features and enhancements added in the Fixed-Point Blockset 3.1 (Release 12.1) since the Fixed-Point Blockset 3.0 (Release 12).

For information about Fixed-Point Blockset features that are incorporated from recent releases, see "New Features" on page 3-2.

This section is organized into these subsections:

New Libraries

The blocks in the Fixed-Point Blockset are now organized into twelve libraries:

- Bits
- Calculus
- Data Type
- Delays & Holds
- Edge Detect
- Filters
- Logic & Comparison
- LookUp
- Math
- Nonlinear
- Select
- Sources

To open the main Fixed-Point Blockset library, which contains the libraries listed above, type `fixpt_lib_3p1` at the MATLAB prompt.

New Blocks

The Fixed-Point Blockset 3.1 includes the new blocks listed below.

Block Name	Library	Description
Accumulator	Calculus	Compute a cumulative sum.
Accumulator Resettable	Calculus	Compute a cumulative sum with external Boolean reset.
Accumulator Resettable Limited	Calculus	Compute a limited cumulative sum with external Boolean reset.
Bit Clear	Bits	Set the specified bit of the stored integer to zero.
Bit Set	Bits	Set the specified bit of the stored integer to one.
Compare To Constant	Logic & Comparison	Determine if a signal is equal to the specified constant.
Compare To Zero	Logic & Comparison	Determine if a signal is equal to zero.
Cosine	LookUp	Implement a sine wave in fixed-point using a lookup table approach that exploits quarter wave symmetry.
Counter Free	Sources	Count up and overflow back to zero after the maximum value possible is reached for the specified number of bits.
Counter Limited	Sources	Count up, and wrap back to zero after outputting the specified upper limit.
Data Type Duplicate	Data Type	Set all inputs to the same data type.
Dead Zone Dynamic	Nonlinear	Set the input within the bounds to zero.
Decrement Real World	Math	Decrease the real world value of the signal by one

Block Name	Library	Description
Decrement Stored Integer	Math	Decrease the stored value of a signal by one
Decrement Time To Zero	Math	Decrease the real world value of the signal by the sample time, but only to zero.
Decrement To Zero	Math	Decrease the real world value of a signal by one, but only to zero.
Derivative	Calculus	Compute a discrete time derivative.
Detect Change	Edge Detect	Detect a change in a signal's value.
Detect Decrease	Edge Detect	Detect a decrease in a signal's value.
Detect Fall Negative	Edge Detect	Detect a falling edge when the signal's value decreases to a strictly negative value, and its previous value was nonnegative.
Detect Fall Nonpositive	Edge Detect	Detect a falling edge when the signal's value decreases to a nonpositive value, and its previous value was strictly positive.
Detect Increase	Edge Detect	Detect an increase in a signal's value.
Detect Rise Nonnegative	Edge Detect	Detect a rising edge when a signal's value increases to a nonnegative value, and its previous value was strictly negative.
Detect Rise Positive	Edge Detect	Detect a rising edge when a signal's value increases to a strictly positive value, and its previous value was nonpositive.
Difference	Calculus	Calculate the change in a signal over one time step.
Filter Direct Form I	Filters	Implement a Direct Form I realization of a filter.
Filter Direct Form I Time Varying	Filters	Implement a time varying Direct Form I realization of a filter.

Block Name	Library	Description
Filter Direct Form II	Filters	Implement a Direct Form II realization of a filter.
Filter Direct Form II Time Varying	Filters	Implement a time varying Direct Form II realization of a filter.
Filter First Order	Filters	Implement a discrete-time first order filter.
Filter Lead or Lag	Filters	Implement a discrete-time lead or lag filter.
Filter Real Zero	Filters	Implement a discrete time filter that has a real zero and no pole.
Increment Real World	Math	Increase the real world value of the signal by one.
Increment Stored Integer	Math	Increase the stored integer value of a signal by one.
Integrator Backward	Calculus	Perform discrete-time integration of a signal using the backward method.
Integrator Backward Resettable	Calculus	Perform discrete-time integration of a signal using the backward method, with external Boolean reset.
Integrator Backward Resettable Limited	Calculus	Perform discrete-time limited integration of a signal using the backward method, with external Boolean reset.
Integrator Forward	Calculus	Perform discrete-time integration of a signal using the forward method.
Integrator Forward Resettable	Calculus	Perform discrete-time integration of a signal using the forward method, with external Boolean reset.
Integrator Forward Resettable Limited	Calculus	Perform discrete-time limited integration of a signal using the forward method, with external Boolean reset.

Block Name	Library	Description
Integrator Trapezoidal	Calculus	Perform discrete-time integration of a signal using the trapezoidal method.
Integrator Trapezoidal Resettable	Calculus	Perform discrete-time integration of a signal using the trapezoidal method, with external Boolean reset.
Integrator Trapezoidal Resettable Limited	Calculus	Perform discrete-time limited integration of a signal using the trapezoidal method, with external Boolean reset.
Interval Test	Logic & Comparison	Determine if a signal is in a specified interval.
Interval Test Dynamic	Logic & Comparison	Determine if a signal is in a specified interval.
MinMax Running Resettable	Math	Determine the minimum or maximum of a signal over time.
Rate Limiter	Nonlinear	Limit the rising and falling rates of the signal.
Rate Limiter Dynamic	Nonlinear	Limit the rising and falling rates of the signal.
Repeating Sequence Interpolated	Sources	Output a discrete-time sequence and repeat, interpolating between data points
Repeating Sequence Stair	Sources	Output a discrete time sequence and repeat.
Sample Time Multiply	Calculus	Support calculations involving sample time.
Saturation Dynamic	Nonlinear	Bound the range of the input.
Scaling Strip	Data Type	Remove scaling and map to a built in integer.
Sine	LookUp	Implement a sine wave in fixed-point using a lookup table approach that exploits quarter wave symmetry.
State-Space	Filters	Implement discrete-time state space.

Block Name	Library	Description
Unit Delay	Delays & Holds	Delay a signal one sample period.
Unit Delay Enabled	Delays & Holds	Delay a signal one sample period.
Unit Delay Enabled External IC	Delays & Holds	Delay a signal one sample period.
Unit Delay Enabled Resettable	Delays & Holds	Delay a signal one sample period.
Unit Delay Enabled Resettable External IC	Delays & Holds	Delay a signal one sample period.
Unit Delay External IC	Delays & Holds	Delay a signal one sample period.
Unit Delay Resettable	Delays & Holds	Delay a signal one sample period.
Unit Delay Resettable External IC	Delays & Holds	Delay a signal one sample period.
Unit Delay With Preview Enabled	Delays & Holds	Support calculations that have feedback and depend on the current input.
Unit Delay With Preview Enabled Resettable	Delays & Holds	Support calculations that have feedback and depend on the current input.
Unit Delay With Preview Enabled Resettable External RV	Delays & Holds	Support calculations that have feedback and depend on the current input.
Unit Delay With Preview Resettable	Delays & Holds	Support calculations that have feedback and depend on the current input.
Unit Delay With Preview Resettable External RV	Delays & Holds	Support calculations that have feedback and depend on the current input.
Wrap To Zero	Nonlinear	Set output to zero if input is above threshold.

Renamed Blocks

All of the Fixed-Point Blockset blocks have been renamed for Version 3.1. Old models built with Version 3.0 will continue to work in Version 3.1.

Version 3.0 Block Name	Version 3.1 Block Name
FixPt Absolute Value	Abs
FixPt Bitwise Operator	Bitwise Operator
FixPt Constant	Constant
FixPt Conversion	Conversion
FixPt Conversion Inherited	Conversion Inherited
FixPt Data Type Propagation	Data Type Propagation
FixPt Dead Zone	Dead Zone
FixPt Dot Product	Dot Product
FixPt Dynamic Look-Up Table	Look-Up Table Dynamic
FixPt FIR	FIR
FixPt Gain	Gain
FixPt Gateway In	Gateway In
FixPt Gateway In Inherited	Gateway In Inherited
FixPt Gateway Out	Gateway Out
FixPt Integer Delay	Integer Delay
FixPt Logical Operator	Logical Operator
FixPt Look-Up Table	Look-Up Table
FixPt Look-Up Table (2D)	Look-Up Table (2D)
FixPt Matrix Gain	Matrix Gain
FixPt MinMax	MinMax

Version 3.0 Block Name	Version 3.1 Block Name
FixPt Multiport Switch	Multiport Switch
FixPt Product	Product
FixPt Relational Operator	Relational Operator
FixPt Relay	Relay
FixPt Saturation	Saturation
FixPt Sign	Sign
FixPt Sum	Sum
FixPt Switch	Switch
FixPt Tapped Delay	Tapped Delay
FixPt Unary Minus	Unary Minus
FixPt Unit Delay	Unit Delay
FixPt Zero-Order Hold	Zero-Order Hold

New Functions

The Fixed-Point Blockset 3.1 includes the new functions listed below.

Function Name	Description
`fixpt_look1_func_approx`	Optimize for a fixed-point function, the x values, or breakpoints, that are generated from a lookup table.
`fixpt_look1_func_plot`	Plot a function with x values generated by the `fixpt_look1_func_approx` function.

Data Type Support

All the Fixed-Point Blockset 3.1 now accept matrix and frame-based signals, except for the following two blocks, which accept only sample-based signals:

- Dot Product
- FIR

If you want to update a model built with Version 3.0 to take advantage of the additional data type support, you should replace all the old blocks in the model with the corresponding Version 3.1 blocks. The section “Renamed Blocks” on page 2-8 lists the Version 3.0 blocks and their corresponding Version 3.1 blocks.

Changes to Functionality

The following changes have been made to the functionality of the Fixed-Point Blockset 3.1 since Version 3.0:

- The **Doubles Override** parameter in the Fixed-Point Blockset Interface has been renamed to **Datatype Override**, and now includes more options.
- The MultiPort Switch block has a new parameter called **Use zero based indexing**.
- The Switch block has a new parameter called **Criteria for passing first input**, which enables you to control the conditions under which the first input is passed.

Known Documentation Problem

This section updates the Fixed-Point Blockset 3.1 documentation set, reflecting a known Fixed-Point Blockset 3.1 documentation problem.

Error in Printed Version of Documentation

The Fixed-Point Blockset 3.1 supports floating-point types, except for custom floating-point types.

In Appendix A, "Code Generation" of the *Fixed-Point Blockset User's Guide*, the section "Storage Class of Variables," in "Code Generation Support," contains the following incorrect reference to data type support in the blockset:

- No floating-point support except for the fixed-point gateway blocks.

This should be corrected as follows:

- Floating-point types are supported, except for custom floating-point types.

3

Fixed-Point Blockset 3.0 Release Notes

New Features

This section introduces the new features and enhancements added in the Fixed-Point Blockset 3.0 since the Fixed-Point Blockset 2.0 (Release 11.0).

This section is organized into these subsections:

New Blocks

The Fixed-Point Blockset 3.0 includes the new blocks listed below.

Block Name	Description
Abs	Output the absolute value of the input.
Bitwise Operator	Perform the specified bitwise operation on the inputs.
Data Type Propagation	Configure the data type and scaling of the propagated signal based on information from the reference signals.
Dead Zone	Provide a region of zero output.
Dot Product	Generate the dot product.
Gateway In Inherited	Convert a Simulink data type to a Fixed-Point Blockset data type, and inherit the data type and scaling.

Block Name	Description
Integer Delay	Delay a signal N sample periods.
Look-Up Table Dynamic	Approximate a one-dimensional function using a selected look-up method and a dynamically specified table.
MinMax	Output the minimum or maximum input value.
Multiport Switch	Switch output between different inputs based on the value of the first input.
Sign	Indicate the sign of the input.
Tapped Delay	Delay a scalar signal multiple sample periods, and output all the delayed versions.
Unary Minus	Negate the input.

New Functions

The Fixed-Point Blockset 3.0 includes the new functions listed below.

Function Name	Description
`fixptbestexp`	Determine the exponent that gives the best precision fixed-point representation of a value.
`fixptbestprec`	Determine the maximum precision available for the fixed-point representation of a value.
`fixpt_convert`	Convert Simulink models and subsystems to fixed-point equivalents.
`fixpt_convert_prep`	Prepare a Simulink model for more complete conversion to fixed point.
`fixpt_restore_links`	Restore links for fixed-point blocks.

Data Type Support

The Fixed-Point Blockset 3.0 has expanded its data type support:

- You can connect Simulink blocks directly to Fixed-Point Blockset blocks provided the signals use built-in Simulink data types. However, a fixed-point signal consisting of 8-, 16-, or 32-bit integers is compatible with Simulink only when its scaling is given by a slope of `1` and a bias of `0`.
- Many important Simulink blocks have added support for fixed-point signals. Key blocks include Enable, Trigger, Scope, Display, and To Workspace.
- Fixed-point blocks now support complex numbers.

Data Type and Scaling Inheritance

Many fixed-point blocks support the inheritance of data type and scaling information from other blocks. The two possible inheritance options are:

- Inherit via internal rule—The output data type and scaling are inherited from the block input(s). The goal of the inheritance rule is to select the "natural" data type and scaling for the output. The specific rule that is used depends on the block operation. For example, if you are multiplying two signed 16-bit signals, the Product block produces the natural output of a signed 32-bit data type.
- Inherit via back propagation—The output data type and scaling are inherited by back propagation. In many cases, you will use the Data Type Propagation block with this option.

Increased Speed, Efficiency, and Accuracy

- Simulation speed has been increased by approximately 2.5 times.
- On average, fixed-point signals use 20 times less memory.
- Interpolation is more accurate and the number of quantization steps has been reduced.
- Generated code is significantly simpler.
- Unsigned numbers avoid overflow in these situations:
  - Interpolation when the slope of the output data is negative
  - Conversion when the bias adjustment is negative
  - Addition and subtraction when the bias adjustment is negative

Overflow Logging

Indirect logging of overflows and saturations via percent range error has been replaced by explicit logging.

Simulink Accelerator

You can now use the Simulink Accelerator in conjunction with the Fixed-Point Blockset. However, you cannot log minimum and maximum simulation values or overflows.

Model Parameter Configuration

You can now use the **Model Parameter Configuration** dialog box to override the **Inline parameters** option for selected parameters. You access this feature by selecting the **Configure** button on the **Advanced** tab of the **Simulation Parameters** dialog box. Previously, code generation for fixed point blocks required all parameters to be inline, or all parameters to be tunable.

Display of Scaling Information

To significantly improve simulation speed, the scaling information associated with fixed-point blocks is no longer updated. Since this scaling information can become obsolete, you should remove it by running `fixpt_clear_tag` or `slupdate`. To view data type and scaling information, you should select **Port data types** under the model's **Format** menu.