Real-Time Workshop®
Embedded Coder

For Use with Real-Time Workshop®

- Modeling
- Simulation
- Implementation

Reference

Version 4

The MathWorks

How to Contact The 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.

Real-Time Workshop Embedded Coder Reference

© COPYRIGHT 2006 by The MathWorks, Inc.

The software described in this document is furnished under a license agreement. The software may be used or copied only under the terms of the license agreement. No part of this manual may be photocopied or reproduced in any form without prior written consent from The MathWorks, Inc.

FEDERAL ACQUISITION: This provision applies to all acquisitions of the Program and Documentation by, for, or through the federal government of the United States. By accepting delivery of the Program or Documentation, the government hereby agrees that this software or documentation qualifies as commercial computer software or commercial computer software documentation as such terms are used or defined in FAR 12.212, DFARS Part 227.72, and DFARS 252.227-7014. Accordingly, the terms and conditions of this Agreement and only those rights specified in this Agreement, shall pertain to and govern the use, modification, reproduction, release, performance, display, and disclosure of the Program and Documentation by the federal government (or other entity acquiring for or through the federal government) and shall supersede any conflicting contractual terms or conditions. If this License fails to meet the government's needs or is inconsistent in any respect with federal procurement law, the government agrees to return the Program and Documentation, unused, to The MathWorks, Inc.

Trademarks

MATLAB, Simulink, Stateflow, Handle Graphics, Real-Time Workshop, and xPC TargetBox are registered trademarks, and SimBiology, SimEvents, and SimHydraulics are trademarks of The MathWorks, Inc.

Other product or brand names are trademarks or registered trademarks of their respective holders.

Patents

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

Revision History

September 2006	Online only	New for Version 4.5 (Release 2006b)

Contents

Functions — By Category

Model Entry Points (p. 1-2)	Access entry points in generated code for Simulink® model
System Target File Callback Interface (p. 1-3)	Control Real-Time Workshop® configuration options in callbacks for ERT-based custom targets

Model Entry Points

`model_initialize`	Initialization entry point in generated code for Simulink model
`model_SetEventsForThisBaseStep`	Set event flags for multirate, multitasking operation before calling *`model`*`_step` for Simulink model
`model_step`	Step routine entry point in generated code for Simulink model
`model_terminate`	Termination entry point in generated code for Simulink model

System Target File Callback Interface

`slConfigUIGetVal`	Return current value for custom target configuration option
`slConfigUISetEnabled`	Enable or disable custom target configuration option
`slConfigUISetVal`	Set value for custom target configuration option

2

Functions — Alphabetical List

model_initialize

Purpose Initialization entry point in generated code for Simulink model

Syntax

```
void model_initialize(void)
void model_initialize(boolean_T firstTime)
```

Arguments

`firstTime`

Real-Time Workshop Embedded Coder generates this argument for ERT-based Simulink models only if the `IncludeERTFirstTime` model configuration parameter is set to `on`. Use of the `firstTime` argument will be discontinued in a future release (see the note below).

Specifies value 0 (FALSE) or 1 (TRUE). If `firstTime` equals 1, *`model`*`_initialize` initializes `rtModel` and other data structures private to the model. If `firstTime` equals 0, *`model`*`_initialize` resets the model's states, but does not initialize other data structures. Call *`model`*`_initialize` with `firstTime` set to 0 to reset the model's states at a time greater than start time.

Description The *`model`*`_initialize` function contains all model initialization code. The generated code for a Simulink model calls *`model`*`_initialize` once, at the beginning of model execution.

If the `IncludeERTFirstTime` model configuration parameter is set to `on`, the generated code passes in `firstTime` as `1` (`TRUE`).

Note In a future release, Real-Time Workshop Embedded Coder will no longer use the `firstTime` argument in a model's generated *`model`*`_initialize` function. For more information about the `IncludeERTFirstTime` model configuration parameter and a related target configuration parameter, `ERTFirstTimeCompliant`, see "Configuration Parameter Reference" in the Real-Time Workshop documentation.

See Also

`model_SetEventsForThisBaseStep`, `model_step`, `model_terminate`

“Model Entry Points” in the Real-Time Workshop Embedded Coder documentation

model_SetEventsForThisBaseStep

Purpose

Set event flags for multirate, multitasking operation before calling `model_step` for Simulink model

Syntax

```
void model_SetEventsForThisBaseStep(boolean_T *eventFlags)
void model_SetEventsForThisBaseStep(boolean_T *eventFlags,
RT_MODEL_model *model_M)
```

Arguments

`eventFlags`
: Pointer to the model's event flags array.

`model_M`
: Pointer to the real-time model object. Real-Time Workshop Embedded Coder generates this argument only if **Generate reusable code** is on.

Description

Real-Time Workshop Embedded Coder generates the `model_SetEventsForThisBaseStep` utility function only for multirate, multitasking models.

The `model_SetEventsForThisBaseStep` function maintains model event flags that determine which subrate tasks need to run on a given base rate time step. In a multirate, multitasking application, the program code must call `model_SetEventsForThisBaseStep` before calling the `model_step` function. See "Multirate Multitasking Operation" in the Real-Time Workshop Embedded Coder documentation for further information.

Note The macro `MODEL_SETEVENTS`, defined in the static `ert_main.c` module, provides a way to call `model_SetEventsForThisBaseStep` from a static main program.

See Also

`model_initialize`, `model_step`, `model_terminate`

"Model Entry Points" in the Real-Time Workshop Embedded Coder documentation

Purpose

Step routine entry point in generated code for Simulink model

Syntax

```
void model_step(void)
void model_step(int_T tid)
void model_stepN (void)
```

Arguments

`tid`

Task identifier. Real-Time Workshop Embedded Coder generates this argument only for multirate, single-tasking models.

Calling Interfaces

The *`model`*`_step` default function prototype varies depending on the number of rates in the model and the solver mode, as shown below:

Rates/Solver Mode	Function Prototype
Single-rate/`SingleTasking`	`void model_step(void);`
Multirate/`SingleTasking`	`void model_step(int_T tid);`
Multirate/`MultiTasking` (rate grouping)	`void model_stepN (void);` (*N* is a task identifier)

If you generate reusable, reentrant code for an ERT-based model using the **Generate reusable code** option, the generated code passes the model's root-level inputs and outputs, block states, parameters, and external outputs to *`model`*`_step` using a function prototype that generally resembles the following:

```
void model_step(inport_args, outport_args, BlockIO_arg,
DWork_arg, RT_model_arg);
```

The manner in which the inport and outport arguments are passed is determined by the setting of the **Pass root-level I/O as** parameter, which appears on the **Interface** pane of the Configuration Parameters dialog box or Model Explorer only if **Generate reusable code** is selected.

Description

Real-Time Workshop Embedded Coder generates the *`model`*`_step` function for a Simulink model when the **Single output/update function** configuration option is selected (the default) in the Configuration Parameters dialog box or Model Explorer. *`model`*`_step` contains the output and update code for all blocks in the model.

`model``_step` is designed to be called at interrupt level from `rt_OneStep`, which is assumed to be invoked as a timer ISR. `rt_OneStep` calls *`model`*`_step` to execute processing for one clock period of the model. See "rt_OneStep" in the Real-Time Workshop Embedded Coder documentation for a description of how calls to *`model`*`_step` are generated and scheduled.

Note If the **Single output/update function** configuration option is not selected, Real-Time Workshop Embedded Coder generates the following model entry point functions in place of *`model`*`_step`:

- *`model`*`_output`: Contains the output code for all blocks in the model
- *`model`*`_update`: Contain the update code for all blocks in the model

---

The *`model`*`_step` function computes the current value of all blocks. If logging is enabled, *`model`*`_step` updates logging variables. If the model's stop time is finite, *`model`*`_step` signals the end of execution when the current time equals the stop time.

In cases where a `tid` is passed in, the caller (`rt_OneStep`) assigns each task a `tid`, and *`model`*`_step` uses the `tid` argument to determine which blocks have a sample hit (and, therefore, should execute).

Under any of the following conditions, *`model`*`_step` does not check the current time against the stop time:

- The model's stop time is set to `inf`.
- Logging is disabled.

- The **Terminate function required** option is not selected.

Therefore, if any of these conditions are true, the program runs indefinitely.

See Also

`model_initialize`, `model_SetEventsForThisBaseStep`, `model_terminate`

"Model Entry Points" in the Real-Time Workshop Embedded Coder documentation

model_terminate

Purpose

Termination entry point in generated code for Simulink model

Syntax

`void model_terminate(void)`

Description

Real-Time Workshop Embedded Coder generates the `model_terminate` function for a Simulink model when the **Terminate function required** configuration option is selected (the default) in the Configuration Parameters dialog box or Model Explorer. `model_terminate` contains all model termination code and should be called as part of system shutdown.

When `model_terminate` is called, blocks that have a terminate function execute their terminate code. If logging is enabled, `model_terminate` ends data logging.

The `model_terminate` function should be called only once.

If your application runs indefinitely, you do not need the `model_terminate` function. To suppress the function, clear the **Terminate function required** configuration option in the Configuration Parameters dialog box or Model Explorer.

See Also

`model_initialize`, `model_SetEventsForThisBaseStep`, `model_step`

"Model Entry Points" in the Real-Time Workshop Embedded Coder documentation

Purpose Return current value for custom target configuration option

Syntax `value = slConfigUIGetVal(hDlg, hSrc, 'OptionName')`

Arguments

`hDlg`
: Handle created in the context of a `SelectCallback` function and used by the System Target File Callback Interface functions. Pass this variable but do not set it or use it for any other purpose.

`hSrc`
: Handle created in the context of a `SelectCallback` function and used by the System Target File Callback Interface functions. Pass this variable but do not set it or use it for any other purpose.

`'OptionName'`
: Quoted name of the TLC variable defined for a custom target configuration option.

Returns Current value of the specified option. The data type of the return value depends on the data type of the option.

Description The `slConfigUIGetVal` function is used in the context of a user-written `SelectCallback` function, which is triggered when the custom target is selected in the System Target File Browser in the Configuration Parameters dialog box or Model Explorer. You use `slConfigUIGetVal` to read the current value of a specified target option.

Examples In the following example, the `slConfigUIGetVal` function returns the value of the **Terminate function required** option on the **Real-Time Workshop/Interface** pane of the Configuration Parameters dialog box or Model Explorer.

```
function usertarget_selectcallback(hDlg, hSrc)

  disp(['*** Select callback triggered:', sprintf('\n'), ...
        '  Uncheck and disable "Terminate function required".']);
```

```
disp(['Value of IncludeMdlTerminateFcn was ', ...
      slConfigUIGetVal(hDlg, hSrc, 'IncludeMdlTerminateFcn')]);

slConfigUISetVal(hDlg, hSrc, 'IncludeMdlTerminateFcn', 'off');
slConfigUISetEnabled(hDlg, hSrc, 'IncludeMdlTerminateFcn', false);
```

See Also

`slConfigUISetEnabled`, `slConfigUISetVal`

“Defining and Displaying Custom Target Options” in the Real-Time Workshop Embedded Coder documentation

“Configuration Parameter Reference” in the Real-Time Workshop documentation

Purpose

Enable or disable custom target configuration option

Syntax

```
slConfigUISetEnabled(hDlg, hSrc, 'OptionName', true)
slConfigUISetEnabled(hDlg, hSrc, 'OptionName', false)
```

Arguments

`hDlg`
: Handle created in the context of a `SelectCallback` function and used by the System Target File Callback Interface functions. Pass this variable but do not set it or use it for any other purpose.

`hSrc`
: Handle created in the context of a `SelectCallback` function and used by the System Target File Callback Interface functions. Pass this variable but do not set it or use it for any other purpose.

`'OptionName'`
: Quoted name of the TLC variable defined for a custom target configuration option.

`true`
: Specifies that the option should be enabled.

`false`
: Specifies that the option should be disabled.

Description

The `slConfigUISetEnabled` function is used in the context of a user-written `SelectCallback` function, which is triggered when the custom target is selected in the System Target File Browser in the Configuration Parameters dialog box or Model Explorer. You use `slConfigUISetEnabled` to enable or disable a specified target option.

Examples

In the following example, the `slConfigUISetEnabled` function disables the **Terminate function required** option on the **Real-Time Workshop/Interface** pane of the Configuration Parameters dialog box or Model Explorer.

```
function usertarget_selectcallback(hDlg, hSrc)
```

```
disp(['*** Select callback triggered:', sprintf('\n'), ...
      '  Uncheck and disable "Terminate function required".']);

disp(['Value of IncludeMdlTerminateFcn was ', ...
      slConfigUIGetVal(hDlg, hSrc, 'IncludeMdlTerminateFcn')]);

slConfigUISetVal(hDlg, hSrc, 'IncludeMdlTerminateFcn', 'off');
slConfigUISetEnabled(hDlg, hSrc, 'IncludeMdlTerminateFcn', false);
```

See Also

`slConfigUIGetVal`, `slConfigUISetVal`

"Defining and Displaying Custom Target Options" in the Real-Time Workshop Embedded Coder documentation

"Configuration Parameter Reference" in the Real-Time Workshop documentation

Purpose

Set value for custom target configuration option

Syntax

```
slConfigUISetVal(hDlg, hSrc, 'OptionName', OptionValue)
```

Arguments

`hDlg`
: Handle created in the context of a `SelectCallback` function and used by the System Target File Callback Interface functions. Pass this variable but do not set it or use it for any other purpose.

`hSrc`
: Handle created in the context of a `SelectCallback` function and used by the System Target File Callback Interface functions. Pass this variable but do not set it or use it for any other purpose.

`'OptionName'`
: Quoted name of the TLC variable defined for a custom target configuration option.

`OptionValue`
: Value to be set for the specified option.

Description

The `slConfigUISetVal` function is used in the context of a user-written `SelectCallback` function, which is triggered when the custom target is selected in the System Target File Browser in the Configuration Parameters dialog box or Model Explorer. You use `slConfigUISetVal` to set the value of a specified target option.

Examples

In the following example, the `slConfigUISetVal` function sets the value `'off'` for the **Terminate function required** option on the **Real-Time Workshop/Interface** pane of the Configuration Parameters dialog box or Model Explorer.

```
function usertarget_selectcallback(hDlg, hSrc)

  disp(['*** Select callback triggered:', sprintf('\n'), ...
       '  Uncheck and disable "Terminate function required".']);

  disp(['Value of IncludeMdlTerminateFcn was ', ...
            slConfigUIGetVal(hDlg, hSrc, 'IncludeMdlTerminateFcn')]);

      slConfigUISetVal(hDlg, hSrc, 'IncludeMdlTerminateFcn', 'off');
      slConfigUISetEnabled(hDlg, hSrc, 'IncludeMdlTerminateFcn', false);
```

See Also

`slConfigUIGetVal`, `slConfigUISetEnabled`

"Defining and Displaying Custom Target Options" in the Real-Time Workshop Embedded Coder documentation

"Configuration Parameter Reference" in the Real-Time Workshop documentation

Blocks — By Category

Configuration Wizards

Custom M-file	Automatically update active configuration parameters of parent model using custom M-file
ERT (optimized for fixed-point)	Automatically update active configuration parameters of parent model for ERT fixed-point code generation
ERT (optimized for floating-point)	Automatically update active configuration parameters of parent model for ERT floating-point code generation
GRT (debug for fixed/floating-point)	Automatically update active configuration parameters of parent model for GRT fixed- or floating-point code generation with debugging enabled
GRT (optimized for fixed/floating-point)	Automatically update active configuration parameters of parent model for GRT fixed- or floating-point code generation

Module Packaging

Data Object Wizard	Simulink data object wizard for creating potential Simulink data objects

Blocks — Alphabetical List

Custom M-file

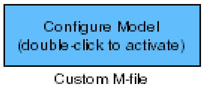
Purpose

Automatically update active configuration parameters of parent model using custom M-file

Library

Configuration Wizards

Description



When you add a Custom M-file block to your Simulink model and double-click it, a custom M-file script executes and automatically configures model parameters that are relevant to code generation. You can also set a block option to invoke the build process after configuring the model.

After double-clicking the block, you can verify that the model parameter values have changed by opening the Configuration Parameters dialog box or Model Explorer and examining the settings.

The MathWorks provides an example M-file script, `*matlabroot*/toolbox/rtw/rtw/rtwsampleconfig.m`, that you can use with the Custom M-file block and adapt to your model requirements. The block and the script provide a starting point for customization. For more information, see "Creating a Custom Configuration Wizard Block" in the Real-Time Workshop Embedded Coder documentation.

Note You can include more than one Configuration Wizard block in your model. This provides a quick way to switch between configurations.

Parameters

Configure the model for

Value selected from

- `ERT (optimized for fixed-point)`
- `ERT (optimized for floating-point)`
- `GRT (optimized for fixed/floating-point)`
- `GRT (debug for fixed/floating-point)`
- `Custom`

For this block, `Custom` is selected by default.

Configuration function

Name of the predefined or custom M-file script to be used to update the active configuration parameters of the parent Simulink model. The default value is `rtwsampleconfig`, which refers to the example M-file script `rtwsampleconfig.m`.

Invoke build process after configuration

If selected, the script initiates the code generation and build process after updating the model's configuration parameters. If not selected (the default), the build process is not initiated.

See Also

ERT (optimized for fixed-point), ERT (optimized for floating-point), GRT (debug for fixed/floating-point), GRT (optimized for fixed/floating-point)

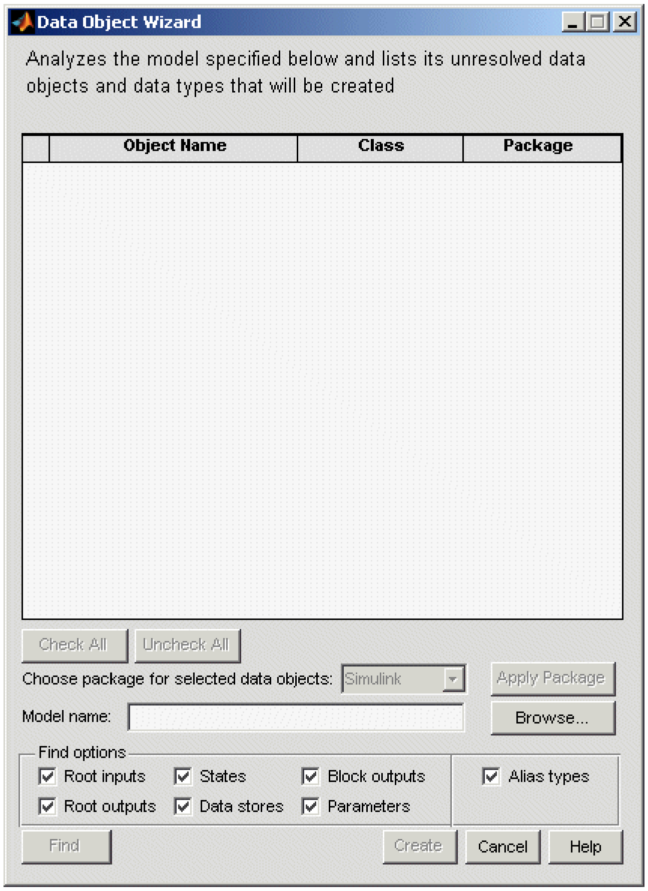
"Optimizing Your Model with Configuration Wizard Blocks and Scripts" in the Real-Time Workshop Embedded Coder documentation

Data Object Wizard

Purpose Simulink data object wizard for creating potential Simulink data objects

Library Module Packaging

Description When you add a Data Object Wizard block to your Simulink model and double-click it, the Data Object Wizard is launched:



Analyzes the model specified below and lists its unresolved data objects and data types that will be created

	Object Name	Class	Package

Choose package for selected data objects: Simulink

Model name:

Find options

- [✓] Root inputs
- [✓] States
- [✓] Block outputs
- [✓] Alias types
- [✓] Root outputs
- [✓] Data stores
- [✓] Parameters

Data Object Wizard

The Data Object Wizard allows you to determine quickly which model data is not associated with Simulink data objects and to create and associate data objects with the data.

For detailed information about using the Data Object Wizard, see "Data Object Wizard" in the Simulink documentation and "Creating Data Objects with Data Object Wizard" in the Real-Time Workshop Embedded Coder documentation.

You can also launch the Data Object Wizard by entering `dataobjectwizard` at the MATLAB® command line or by selecting **Data Object Wizard** from the **Tools** menu of your model.

Example

For an example of a model that incorporates the Data Object Wizard block, see `rtwdemo_mpf`.

See Also

"Data Object Wizard" in the Simulink documentation

"Creating Data Objects with Data Object Wizard" in the Real-Time Workshop Embedded Coder documentation

"Creating a Data Dictionary for a Model" in the Real-Time Workshop Embedded Coder documentation

"Customizing Data Object Wizard User Packages" in the Real-Time Workshop Embedded Coder documentation

ERT (optimized for fixed-point)

Purpose

Automatically update active configuration parameters of parent model for ERT fixed-point code generation

Library

Configuration Wizards

Description

ERT (optimized for fixed-point)

When you add an ERT (optimized for fixed-point) block to your Simulink model and double-click it, a predefined M-file script executes and automatically configures the model parameters optimally for fixed-point code generation with the ERT target. You can also set a block option to invoke the build process after configuring the model.

After double-clicking the block, you can verify that the model parameter values have changed by opening the Configuration Parameters dialog box or Model Explorer and examining the settings.

Note You can include more than one Configuration Wizard block in your model. This provides a quick way to switch between configurations.

Parameters

Configure the model for

Value selected from

- `ERT (optimized for fixed-point)`
- `ERT (optimized for floating-point)`
- `GRT (optimized for fixed/floating-point)`
- `GRT (debug for fixed/floating-point)`
- `Custom`

For this block, `ERT (optimized for fixed-point)` is selected by default.

Configuration function

Grayed out unless **Configure the model for** is set to `Custom`. This parameter is used with the Custom M-file block.

Invoke build process after configuration

If selected, the script initiates the code generation and build process after updating the model's configuration parameters. If not selected (the default), the build process is not initiated.

See Also

Custom M-file, ERT (optimized for floating-point), GRT (debug for fixed/floating-point), GRT (optimized for fixed/floating-point)

"Optimizing Your Model with Configuration Wizard Blocks and Scripts" in the Real-Time Workshop Embedded Coder documentation

ERT (optimized for floating-point)

Purpose Automatically update active configuration parameters of parent model for ERT floating-point code generation

Library Configuration Wizards

Description When you add an ERT (optimized for floating-point) block to your Simulink model and double-click it, a predefined M-file script executes and automatically configures the model parameters optimally for floating-point code generation with the ERT target. You can also set a block option to invoke the build process after configuring the model.

After double-clicking the block, you can verify that the model parameter values have changed by opening the Configuration Parameters dialog box or Model Explorer and examining the settings.

Note You can include more than one Configuration Wizard block in your model. This provides a quick way to switch between configurations.

Parameters

Configure the model for

Value selected from

- `ERT (optimized for fixed-point)`
- `ERT (optimized for floating-point)`
- `GRT (optimized for fixed/floating-point)`
- `GRT (debug for fixed/floating-point)`
- `Custom`

For this block, `ERT (optimized for floating-point)` is selected by default.

Configuration function

Grayed out unless **Configure the model for** is set to `Custom`. This parameter is used with the Custom M-file block.

Invoke build process after configuration
If selected, the script initiates the code generation and build process after updating the model's configuration parameters. If not selected (the default), the build process is not initiated.

See Also

Custom M-file, ERT (optimized for fixed-point), GRT (debug for fixed/floating-point), GRT (optimized for fixed/floating-point)

"Optimizing Your Model with Configuration Wizard Blocks and Scripts" in the Real-Time Workshop Embedded Coder documentation

GRT (debug for fixed/floating-point)

Purpose

Automatically update active configuration parameters of parent model for GRT fixed- or floating-point code generation with debugging enabled

Library

Configuration Wizards

Description

When you add a GRT (debug for fixed/floating-point) block to your Simulink model and double-click it, a predefined M-file script executes and automatically configures the model parameters optimally for fixed/floating-point code generation, with TLC debugging options enabled, with the GRT target. You can also set a block option to invoke the build process after configuring the model.

After double-clicking the block, you can verify that the model parameter values have changed by opening the Configuration Parameters dialog box or Model Explorer and examining the settings.

Note You can include more than one Configuration Wizard block in your model. This provides a quick way to switch between configurations.

Parameters

Configure the model for

Value selected from

- `ERT (optimized for fixed-point)`
- `ERT (optimized for floating-point)`
- `GRT (optimized for fixed/floating-point)`
- `GRT (debug for fixed/floating-point)`
- `Custom`

For this block, `GRT (debug for fixed/floating-point)` is selected by default.

Configuration function

Grayed out unless **Configure the model for** is set to `Custom`. This parameter is used with the Custom M-file block.

Invoke build process after configuration
If selected, the script initiates the code generation and build process after updating the model's configuration parameters. If not selected (the default), the build process is not initiated.

See Also

Custom M-file, ERT (optimized for fixed-point), ERT (optimized for floating-point), GRT (optimized for fixed/floating-point)

"Optimizing Your Model with Configuration Wizard Blocks and Scripts" in the Real-Time Workshop Embedded Coder documentation

GRT (optimized for fixed/floating-point)

Purpose

Automatically update active configuration parameters of parent model for GRT fixed- or floating-point code generation

Library

Configuration Wizards

Description

When you add a GRT (optimized for fixed/floating-point) block to your Simulink model and double-click it, a predefined M-file script executes and automatically configures the model parameters optimally for fixed/floating-point code generation with the GRT target. You can also set a block option to invoke the build process after configuring the model.

After double-clicking the block, you can verify that the model parameter values have changed by opening the Configuration Parameters dialog box or Model Explorer and examining the settings.

Note You can include more than one Configuration Wizard block in your model. This provides a quick way to switch between configurations.

Parameters

Configure the model for

Value selected from

- `ERT (optimized for fixed-point)`
- `ERT (optimized for floating-point)`
- `GRT (optimized for fixed/floating-point)`
- `GRT (debug for fixed/floating-point)`
- `Custom`

For this block, `GRT (optimized for fixed/floating-point)` is selected by default.

Configuration function

Grayed out unless **Configure the model for** is set to `Custom`. This parameter is used with the Custom M-file block.

Invoke build process after configuration
If selected, the script initiates the code generation and build process after updating the model's configuration parameters. If not selected (the default), the build process is not initiated.

See Also

Custom M-file, ERT (optimized for fixed-point), ERT (optimized for floating-point), GRT (debug for fixed/floating-point)

"Optimizing Your Model with Configuration Wizard Blocks and Scripts" in the Real-Time Workshop Embedded Coder documentation

Index