

Latest Features in Embedded Coder

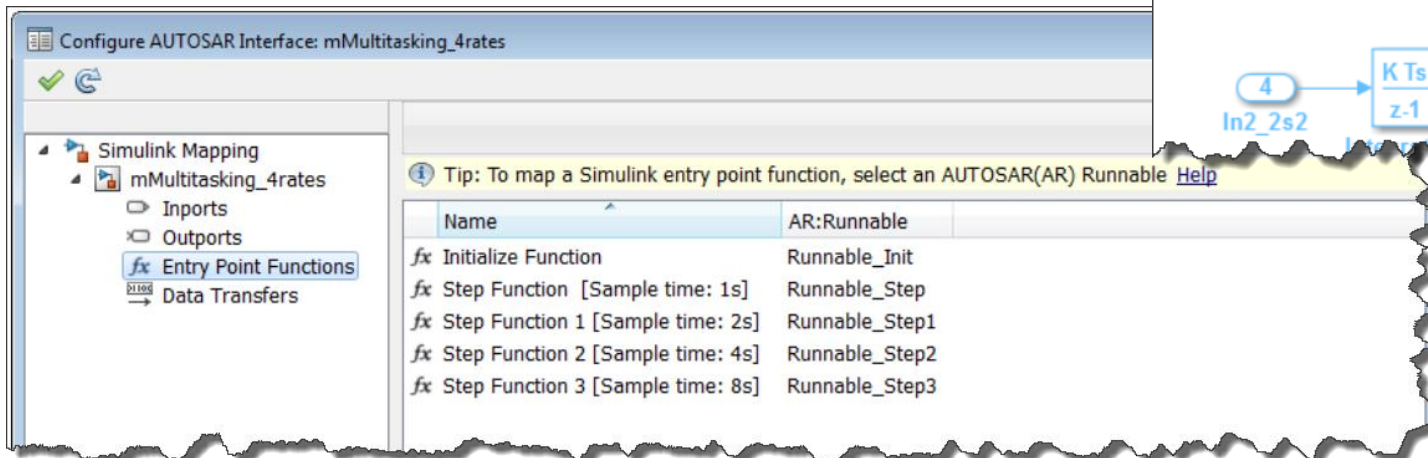
March 2015

R2015a

AUTOSAR Rate-Based Multi-Runnables

Model multiple runnables using a rate-based multitasking modeling style

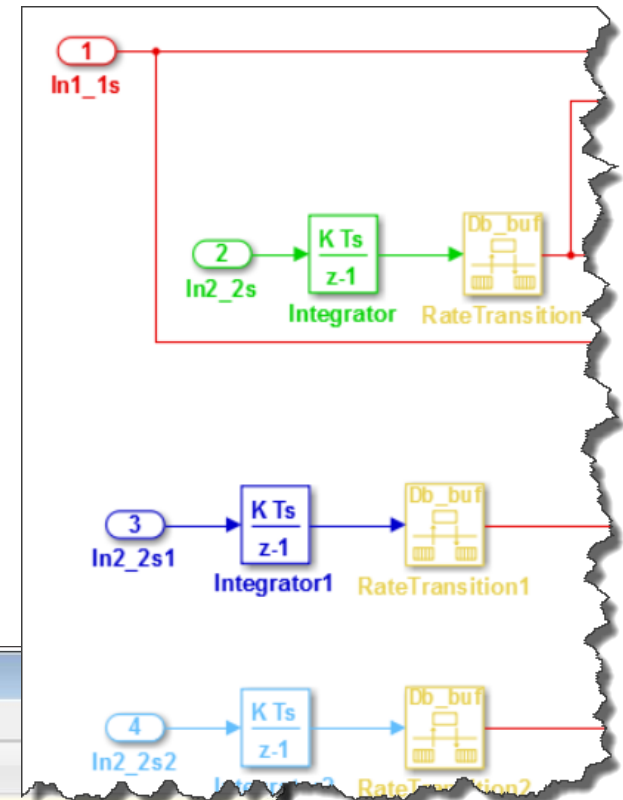
- Use Rate Transition blocks to handle data transfers between blocks at different rates
- Previously, multiple runnables were only supported using a function-call based modeling style



Configure AUTOSAR Interface: mMultitasking_4rates

Tip: To map a Simulink entry point function, select an AUTOSAR(AR) Runnable [Help](#)

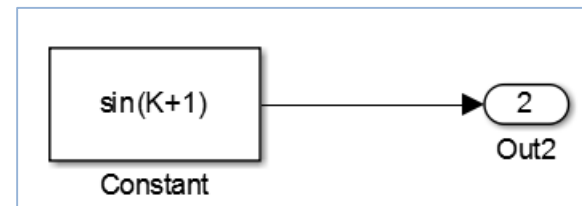
Name	AR:Runnable
fx Initialize Function	Runnable_Init
fx Step Function [Sample time: 1s]	Runnable_Step
fx Step Function 1 [Sample time: 2s]	Runnable_Step1
fx Step Function 2 [Sample time: 4s]	Runnable_Step2
fx Step Function 3 [Sample time: 8s]	Runnable_Step3



Increased Code Efficiency

Improved RAM/ROM and execution speed efficiency for select code patterns

- Model block I/O improvements including reuse of input and output code using reusable custom storage class
- Unit delay block improvements including input, output, and state code reuse
- Global data references replaced with locals to enable expression folding



R2014b

```
/* Outport: '<Root>/Out2' incorporates:
 * Constant: '<Root>/Constant'
 */
fastz1_Y.Out2 = K + 1.0;
fastz1_Y.Out2 = sin(fastz1_Y.Out2);
```

R2015a

```
/* Outport: '<Root>/Out2' incorporates:
 * Constant: '<Root>/Constant'
 */
fastz1_Y.Out2 = sin(K + 1.0);
```

Control Booleans and Data Type Limits

Specify Boolean and data type limits in generated code

- Control Boolean identifiers, which appear by default in `rtwtypes.h`
- Define in separate file and include in `rtwtypes.h` (optional)
- Control macros used to define type limits (e.g., `Max_int16_T`)

Specify Boolean

```
set_param(gcs,'BooleanTrueId','bTrue');  
set_param(gcs,'BooleanFalseId','bFalse')
```

Generated Code (`rtwtypes.h`)

```
#define bFalse (0U)  
#define bTrue (1U)
```

In/Out Function Prototype Control

Combine input and output arguments of step functions

- Ability to specify step function as pointer to reuse argument for in/out
- C function prototype control
- C++ class interface control

Configure model initialize and step functions

Initialize function name:

Step function name:

Step function arguments:

Order	Port Name	Port Type	Category	Argument Name	Qualifier
1	In1	Inport	Value	argIn1	const
2	Out2	Outport	Pointer	arg_Out2	none
3	Out1	Outport	Pointer	sharedArg	none
4	In2	Inport	Pointer	sharedArg	none

Up
Down

Step function preview

model_step_custom (argIn1, * arg_Out2, * sharedArg)

Validation

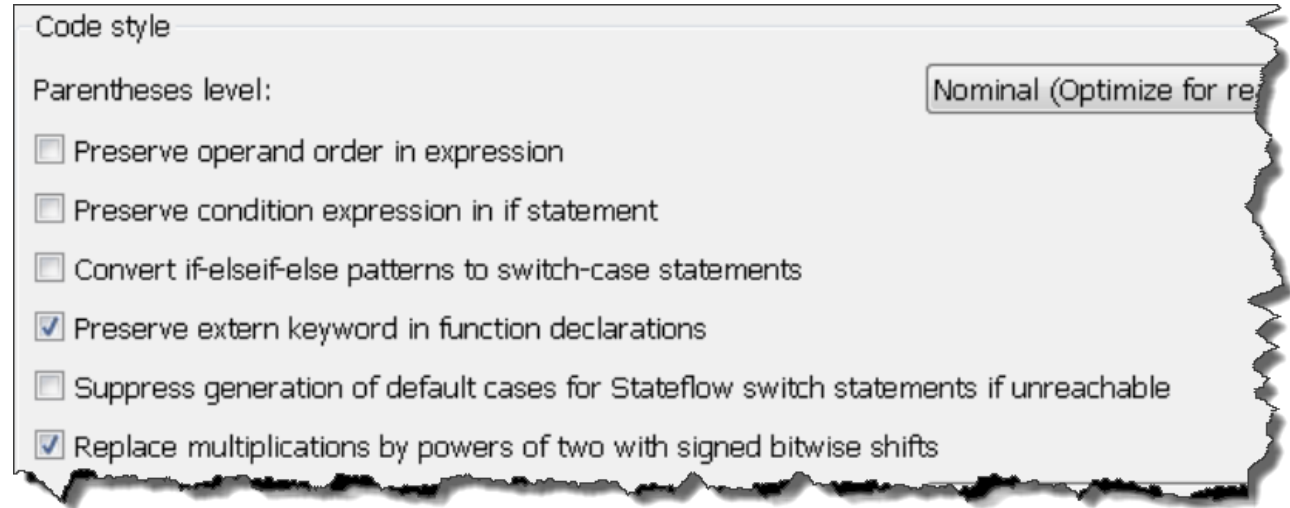
(*invokes update diagram)

✓ Last validation succeeded.

MISRA-C Support for Bitwise Operations on Signed Integers

Replace powers of two multiplication with bitwise shifts

New



☒ On `Y.Out1 = (U.In1 << ((int8_T)3));`

MISRA

☐ Off `Y.Out1 = U.In1 * ((int64_T)8);`