Latest Features in Embedded Coder

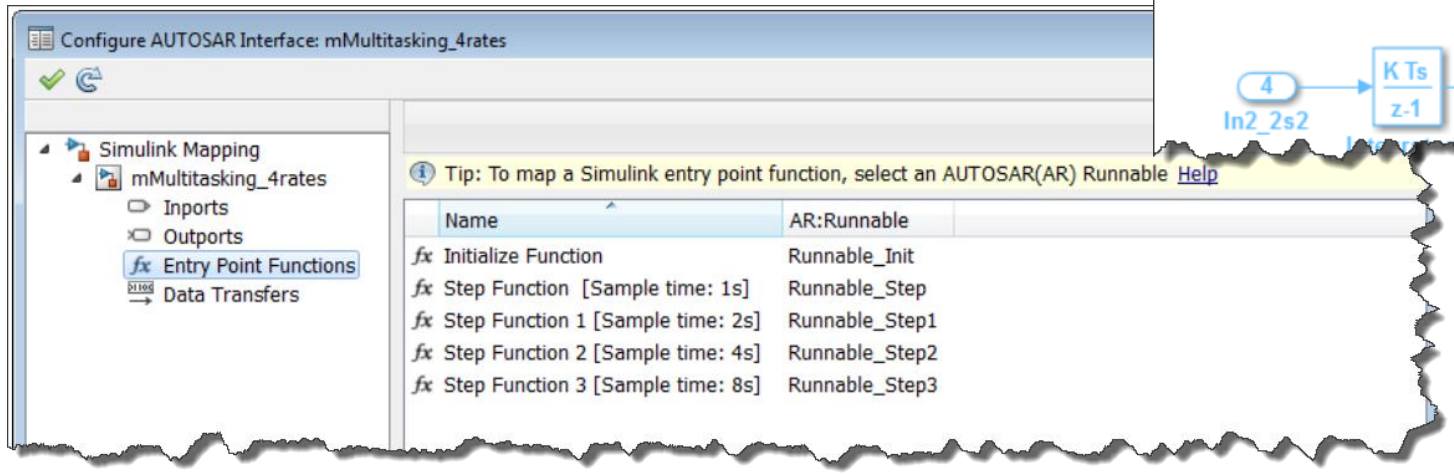
March 2015

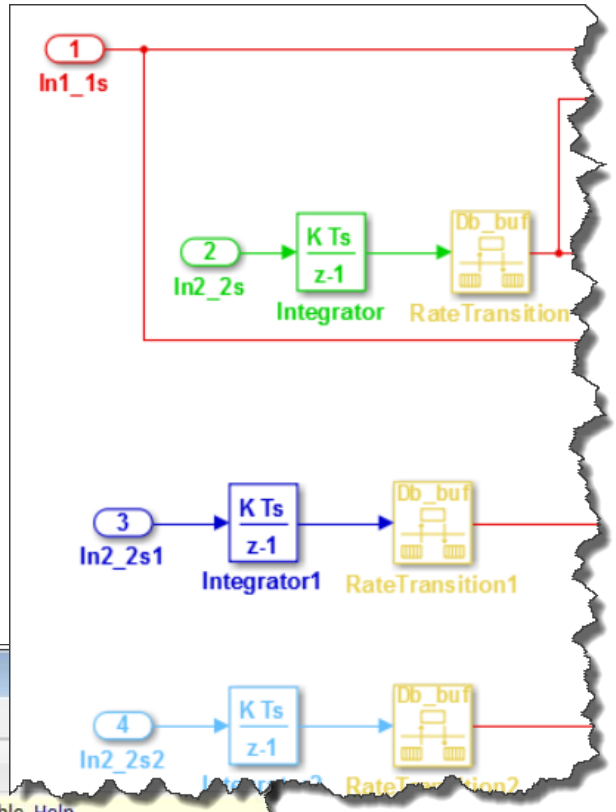
R2015a

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

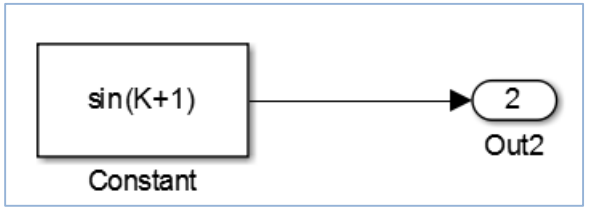




Increased Code Efficiency

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')
```

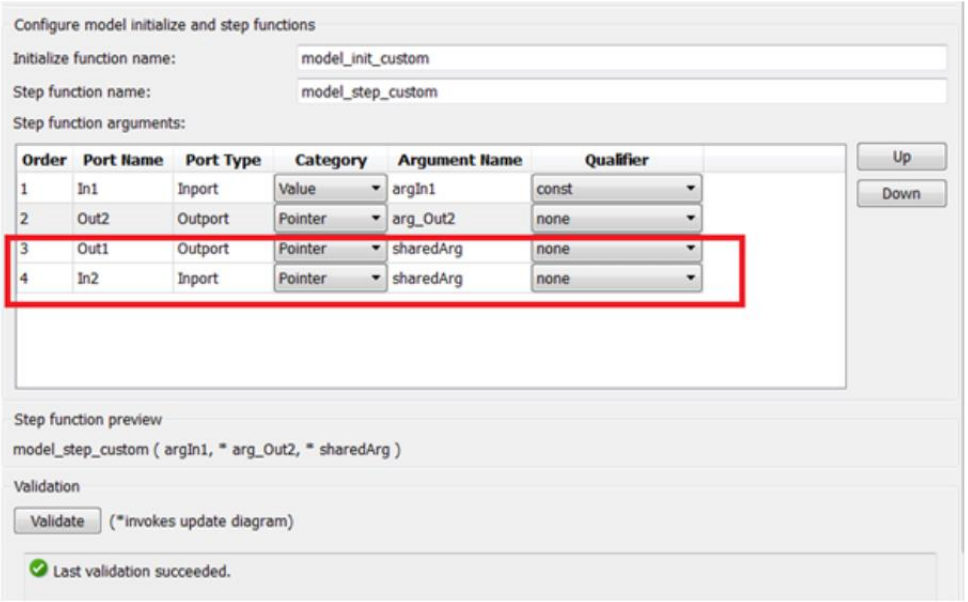
Generated Code (`rtwtypes.h`)

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

In/Out Function Prototype Control

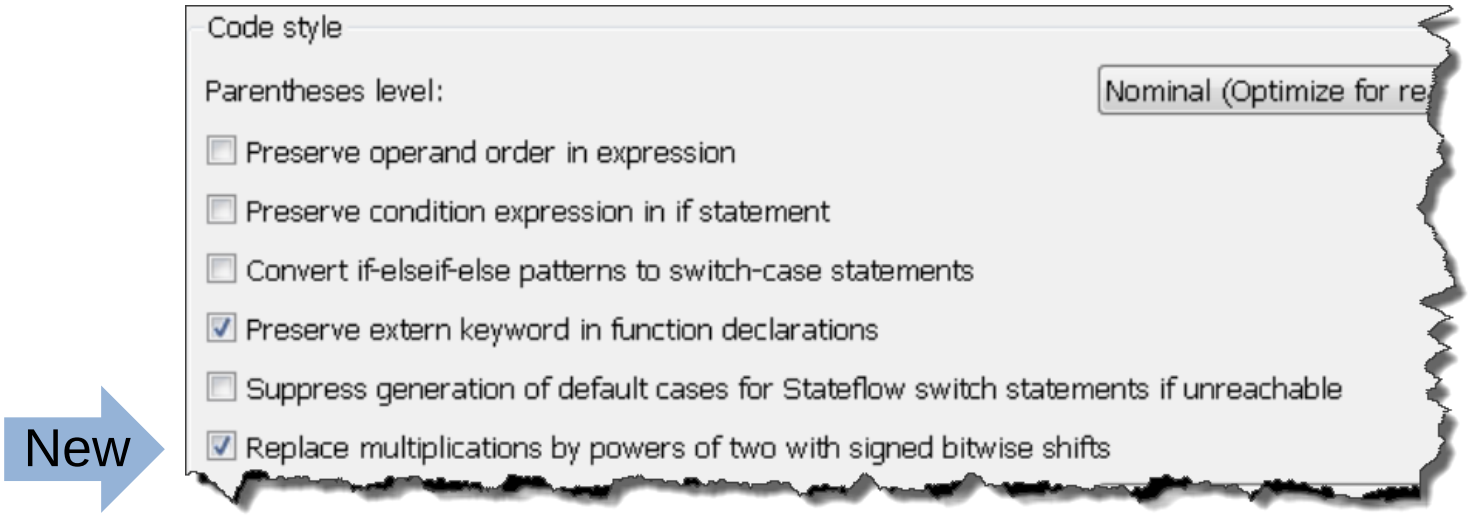
Combine input and output arguments of step functions

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



MISRA-C Support for Bitwise Operations on Signed Integers

Replace powers of two multiplication with bitwise shifts



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

MISRA

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