Latest Features in Embedded Coder

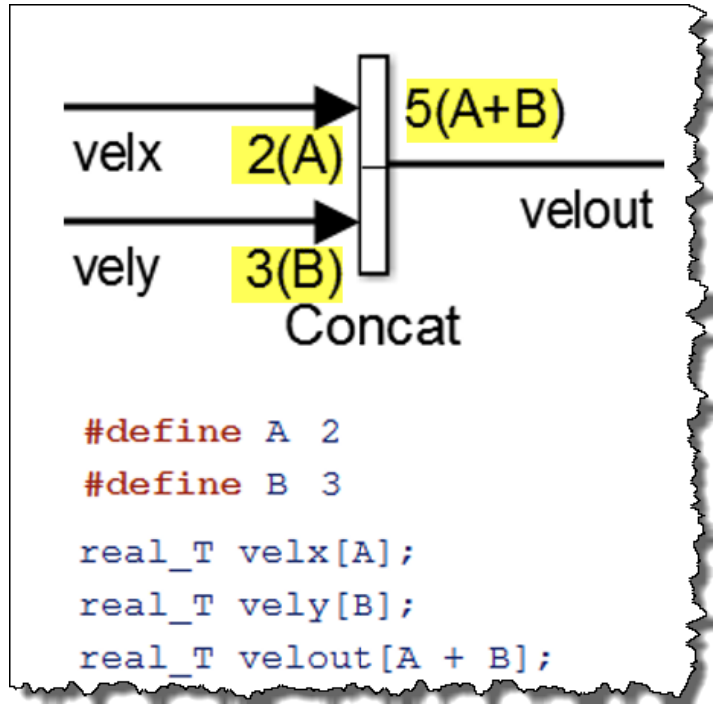
March 2016

R2016a

Compile-Time Dimensions

Generate compiler directives for signal dimensions

- Use `Simulink.Parameter` (as in a MATLAB expression) to represent a dimension value
- Propagate dimension symbols throughout model during simulation

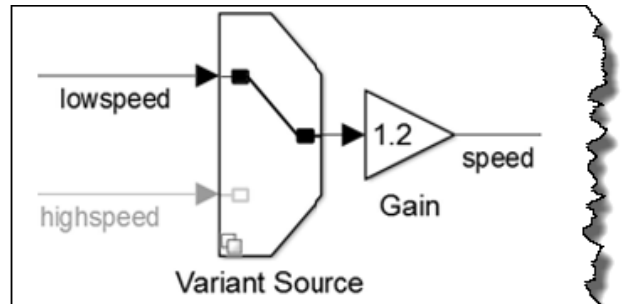


» `rtwdemo_dimension_variants.slx`

Compile-Time Variants

Generate compiler directives based on Variant Source/Sink blocks

- Finely place variant choices within models
- Generate code for only the active variant, or, generate preprocessor conditionals and decide the active variant at compilation time

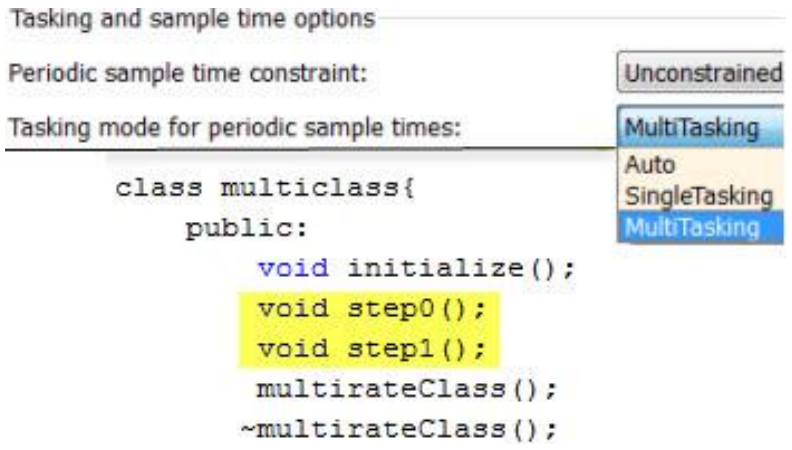



```
#if Type == 1

  VariantMerge = lowspeed;

#endif

#if Type == 2

  VariantMerge = highspeed;

#endif

  speed = 1.2 * VariantMerge;
}
```

Enhanced C++ Code Generation

Use referenced models with multitasking, export-functions, and virtual buses

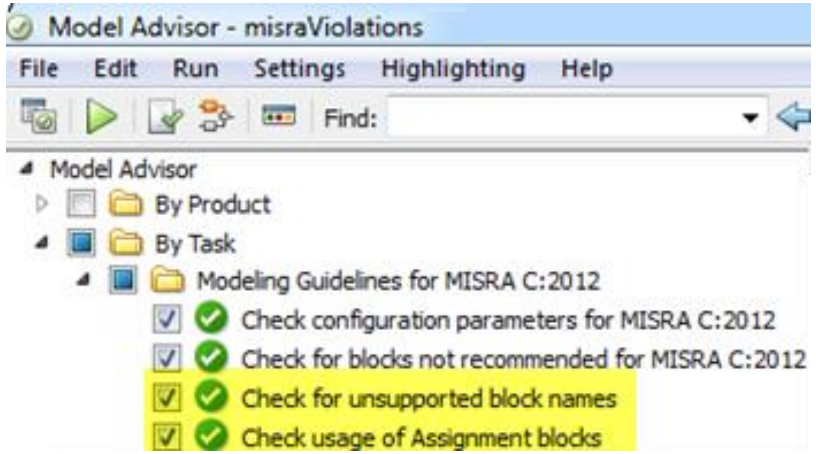
- C++ class interface supports:
  - Multitasking for model references
  - Export function-call subsystems
  - Virtual buses for crossing model boundaries



MISRA C:2012 Compliance

Check block names and Assignment blocks

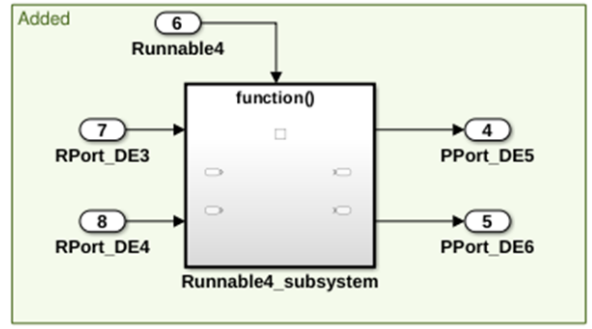
- Use Model Advisor to:
  - Check for block names that contain a / character
  - Check usage of assignment block initializations

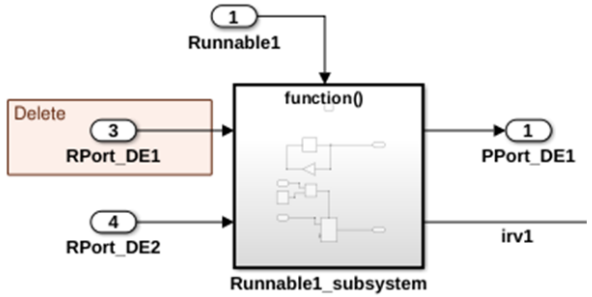


Enhanced AUTOSAR Round-Trip Workflow

Merge AUTOSAR authoring tool changes into Simulink using ARXML files

- Automate Simulink block additions and deletions with highlighting
- Support CompuMethods with LINEAR and TEXTTABLE COMPU-SCALES
- Enhance control of AUTOSAR package path specification

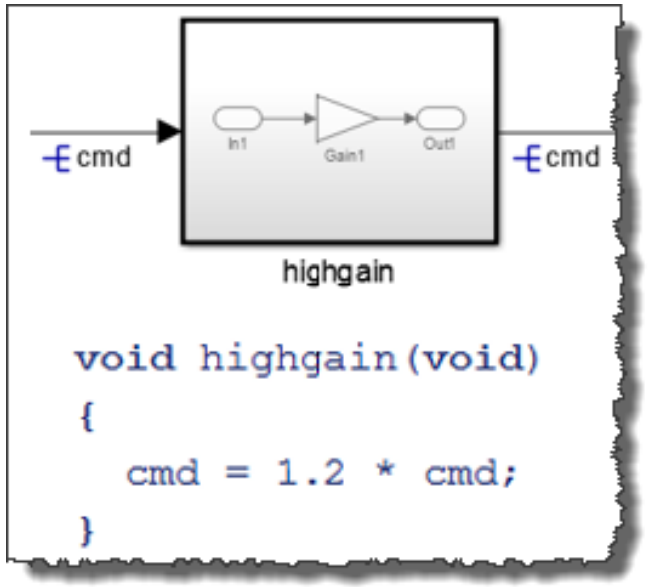




Data Buffer Reuse

Use same variable for signals in a path as input and output

- Specify IN/OUT signals at a block and subsystem boundary, or blocks and subsystems in a path
- Share `Reusable` custom storage class



Buffer Reuse Across Model Blocks

Use same variable for input and output arguments of MATLAB Function and Model blocks

- Previously, there was no buffer reuse across model blocks, and a data copy occurred prior to each model step function
- In R2016a, input and output buffers of Model blocks are reused

R2015b

```
(void) memcpy(&Mdl1A[0], &Add[0], 9*sizeof(real));
 MDLOBJ1.step(&Mdl1A[0], &Add1[0]);
(void) memcpy(&Mdl1B[0], &Mdl1A[0], 9*sizeof(real));
 MDLOBJ2.step(&Mdl1B[0], &Add1[0]);
```

R2016a

```
MDLOBJ1.step(&Mdl1AB[0], &Add1[0]);
MDLOBJ2.step(&Mdl1AB[0], &Add1[0]);
memcpy(&parent.Out1[0], &Mdl1AB[0],
        (uint32_T)(9U * sizeof(real)));
```

Data Access Support for SIL and PIL

Use vector `GetSet` custom storage class and C++ parameter access methods

SIL and PIL support for:

- `GetSet` custom storage class for vector signals and parameters
- Method and Inlined method options

```
double get_inVector(int index)
{
    return ex_getset_data.vectors.inVector[index];
}

void set_inVector(int index, double value)
{
    ex_getset_data.vectors.inVector[index] = value;
}
```

Dual Core TI C2000 Delfino Support

Code generation for Texas Instruments Delfino F2833x, F2837xS/D, and C2834x

- Real-time parameter tuning and logging using external mode
- Perform processor-in-the-loop (PIL) with execution profiling
- Block libraries for on-chip peripherals
- Generate code for both cores of F2837xD