

# Latest Features in Real-Time Workshop Embedded Coder

March 2009

**R2009a**

# Code Generation Objectives

## Challenge

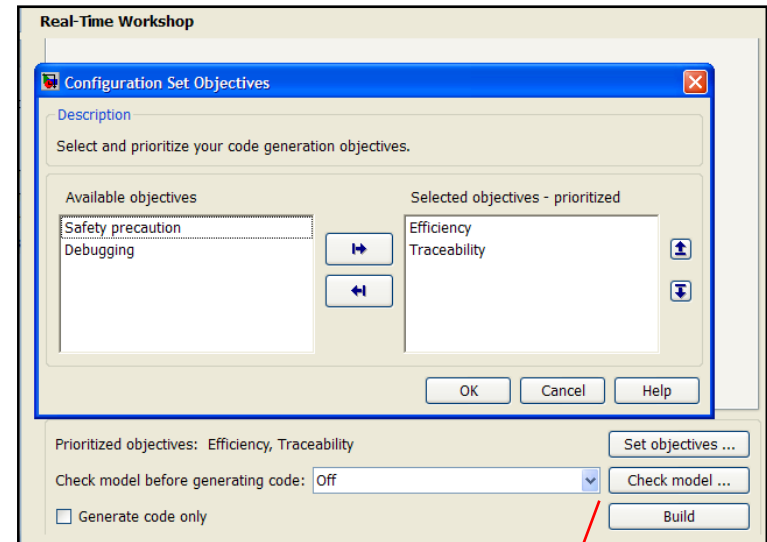
- There are many options to examine for optimal code generation configuration.
- Optimal settings depend on each project's objectives.
- New options are added each release.

## Solution

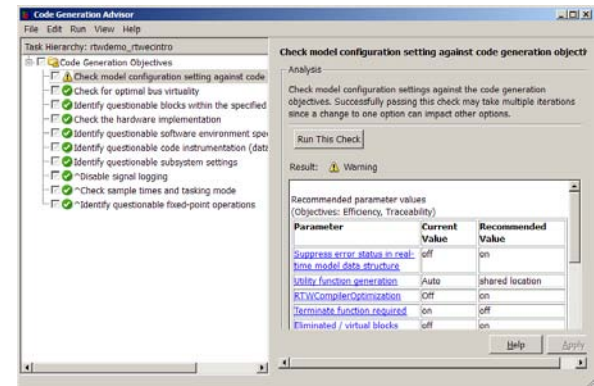
- Allow users to specify and prioritize high-level code generation objectives
- Use Model Advisor to guide users

## Benefit

- Reduced learning curve
- Better model and code settings
- Faster adoption of new options



Objectives-based model checking on demand or tied to build process



# Custom Storage Class Specification on Port

## Challenge

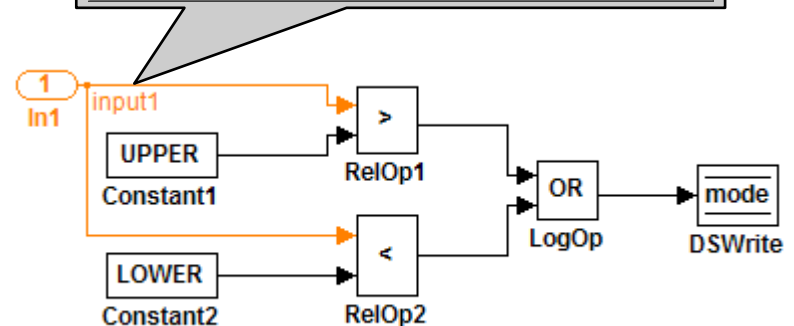
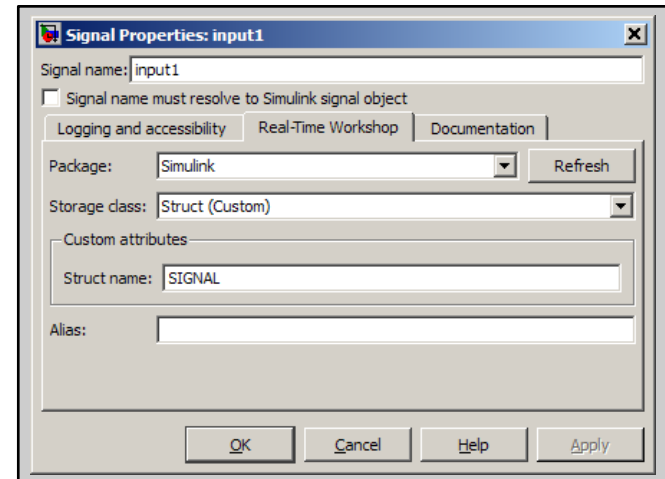
- User had to create Simulink signal objects in the MATLAB workspace to specify Custom Storage Classes (CSCs) for signals.

## Solution

- Allow for specification of CSCs directly on Signal Properties dialog boxes

## Benefit

- Simpler configuration management
- Easier specification of CSCs for signals
- Less clutter in the MATLAB workspace



```
typedef struct SIGNAL_tag {
    MYTYPE input1;
} SIGNAL_type;

SIGNAL_type SIGNAL;
```

Demo: >>rtwdemo\_advsc

# Reduction in Global Data and Data Copies for Reusable Subsystems

## Challenge

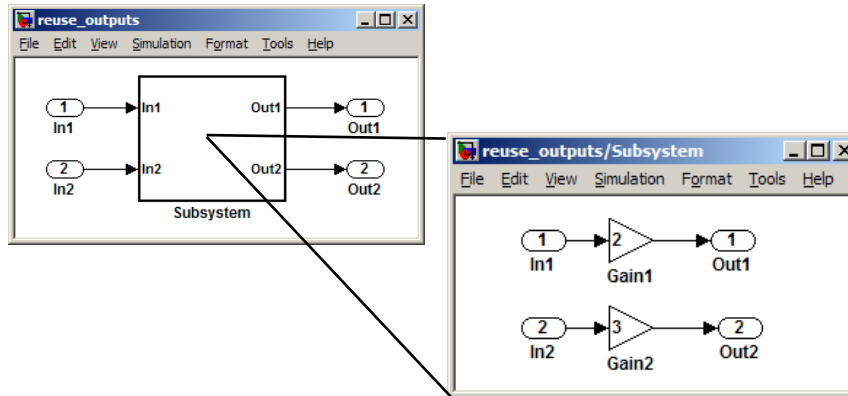
- Reusable subsystem outputs in global data structure can be inefficient.

## Solution

- Provide option to pass subsystem outputs as individual arguments

## Benefit

- For some use cases:
  - Reduces RAM
  - Improves speed



```
void Subsystem(real_T rtu_In1, real_T rtu_In2,
real_T *rtu_0, real_T *rtu_1)
{
(*rtu_0) = 2.0 * rtu_In1;
(*rtu_1) = 3.0 * rtu_In2;
}

void reuse_outputs_step(void)
{
real_T rtb_SubsystemOut1;
real_T rtb_SubsystemOut2;

Subsystem(rtU.In1, rtU.In2, &rtb_SubsystemOut1,
&rtb_SubsystemOut2);
:
}
```

Individual arguments (no global BlockIO)

```
typedef struct {
real_T Gain1;
real_T Gain2;
} rtB_Subsystem
```

```
void Subsystem(real_T rtu_In1, real_T rtu_In2,
rtB_Subsystem *localB)
{
localB->Gain1 = 2.0 * rtu_In1;
localB->Gain2 = 3.0 * rtu_In2;
}

void reuse_outputs_step(void)
{
Subsystem(rtU.In1, rtU.In2, &rtB.Subsystem_c);
:
}
```

Structure reference (uses global BlockIO)

# Selector/Assignment Efficiency Improvement

## Challenge

- Selector/assignment is a common pattern that may have unnecessary data copies.

## Solution

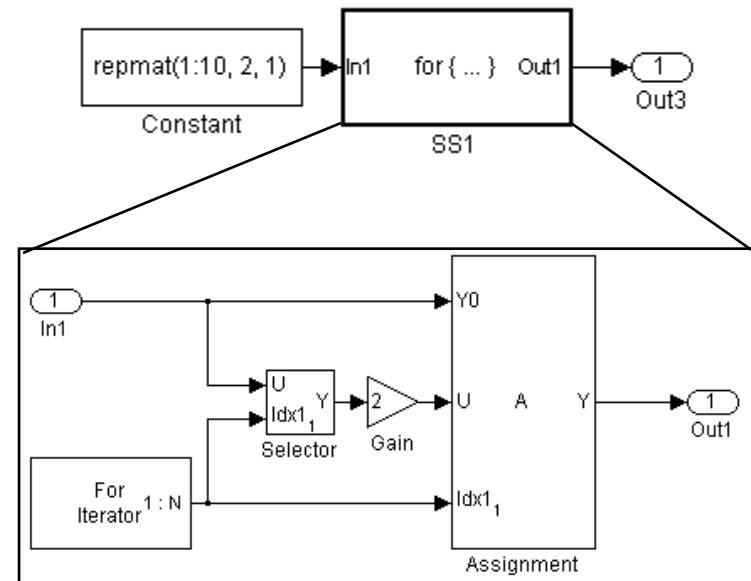
- Optimize code by removing unnecessary loops and data copies

```
i = 0;
for (uDimIdx = 0; uDimIdx < 10; uDimIdx++) {
    rtb_Selector[i] = rtP.Constant[(uDimIdx << 1) + s1_iter];
    i++;
}
```

```
for (i = 0; i < 10; i++) {
    rtb_Gain[i] = rtP.Gain * rtb_Selector[i];
}
```

```
i = 0;
for (uDimIdx = 0; uDimIdx < 10; uDimIdx++) {
    rtY.Out3[(uDimIdx << 1) + s1_iter] = rtb_Gain[i];
    i++;
}
```

R2008b



```
for (tmp = 0; tmp < 10; tmp++) {
    rtY.Out3[j + (tmp << 1)] = rtP.Constant[(tmp << 1) + i] * rtP.Gain;
}
```

R2009a

# Processor-in-the-Loop (PIL) Mode for Model Blocks

## Challenge

- Model block PIL did not support model arguments or global tunable parameters.

## Solution

- Support tuning of model arguments and global tunable parameters during PIL test
- Propagate parameter changes to code executing on the PIL target hardware

## Benefit

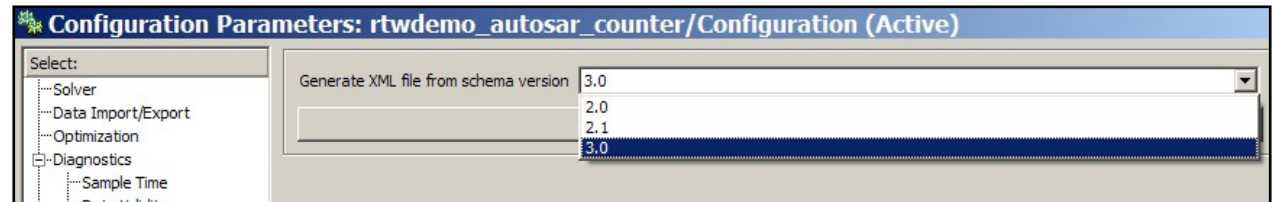
- Improved code verification and tuning capability

The screenshot shows a Simulink model window titled 'rtwdemo\_pil \*'. Inside the model, a block named 'rtwdemo\_pil\_component (PIL)' is connected to a 'counterA' block. The 'rtwdemo\_pil\_component' block has an 'input' and an 'output' port. The 'counterA' block has a 'counterA' input and an 'error' output. A feedback loop is shown with a summing junction and a gain block.

Two 'Function Block Parameters' dialog boxes are overlaid on the model. The top dialog box is for the 'Gain' parameter, showing 'Gain: Element-wise gain (y = K.\*u) or matrix gain (y = K\*u)'. The 'Parameter Attributes' tab is selected, showing 'Gain: Increment' and 'Multiplication: Element-wise(K.\*u)'. A blue callout box points to the 'Gain' field with the text: 'Tunable parameter defined as a Simulink.Parameter in the Base Workspace'.

The bottom dialog box is for the 'Model Reference' parameter, showing 'Model Reference' and 'Parameter' tabs. The 'Model name (without the .mdl extension):' field contains 'rtwdemo\_pil\_component'. The 'Model arguments:' field contains 'myLower, myUpper'. The 'Model argument values (for this instance):' field contains '30, 70'. A blue callout box points to the 'Model arguments' field with the text: 'Tunable model arguments'.

# AUTOSAR v3.0 Support



## Challenge

- AUTOSAR version 3.0 was not supported.

## Solution

- Support import/export of AUTOSAR descriptions using v3.0 of the standard

## Benefit

- AUTOSAR descriptions for all major AUTOSAR schema versions supported

# Encapsulated C++ for Model Reference Hierarchy

## Challenge

- Encapsulated C++ (using classes) was only applied to top level of Model Reference hierarchy.

## Solution

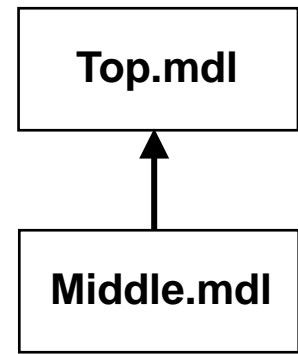
- Generate a class for each node of the hierarchy
- Allow interface specification for each node

## Benefit

- Improve modularity

```

110 class Top {
111     /* public data and function members */
112     public:
113     /* Model entry point functions */
114
115     /* model initialize function */
116     void initialize(void);
117
118     /* model step function */
119
120     ...
130     RT_MODEL top * getRTM(void);
131
132     /* private data and function members */
133     private:
134     BlockIO top top_B;           /* Block
135     D Work top top_DWork;       /* Block
136     RT_MODEL top top_M;        /* Real-t
137     RT_MODEL top *top_M;       /* Real-t
138     Middle Middle_instance_1;  /* model
139 };
    
```



```

18 /* Model step function */
19 void Top::step(real_T arg_upper1, real_T arg_In1, real_T *arg_out
20               *arg_Out1)
21 {
22     /* OutputUpdate for ModelReference Block: '<Root>/Middle' */
23     Middle_instance_1.step(&top_B.Middle_o1, arg_upper1, &top_B.Mi
24                           &top_RGND, &top_RGND, &top_B.Middle_o3, &top_B.Middle_o4);
25
26     /* Outport: '<Root>/output2' */
27     (*arg_output2) = top_B.Middle_o2;
    
```