MathWorks®

Latest Features in Simulink Coder

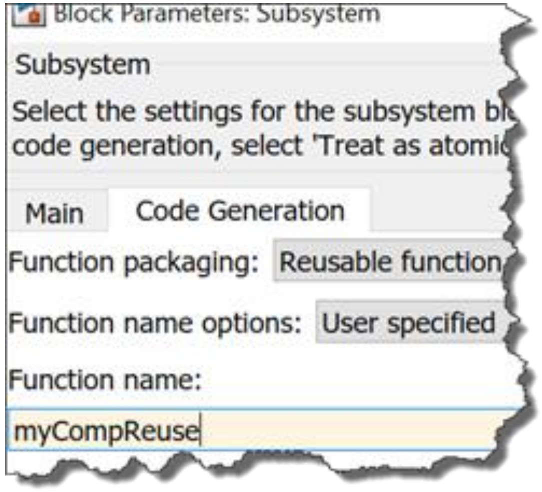
March 2017

R2017a

Subsystem Reuse Across Models

Reuse subsystems with naming control and global Data Store Memory blocks across models

- Generate reusable code for
  - Subsystems across model reference boundaries that contain Data Store blocks that read from or write to a global data store
  - Reusable functions with user-specified names



Block Parameters: Subsystem

Subsystem

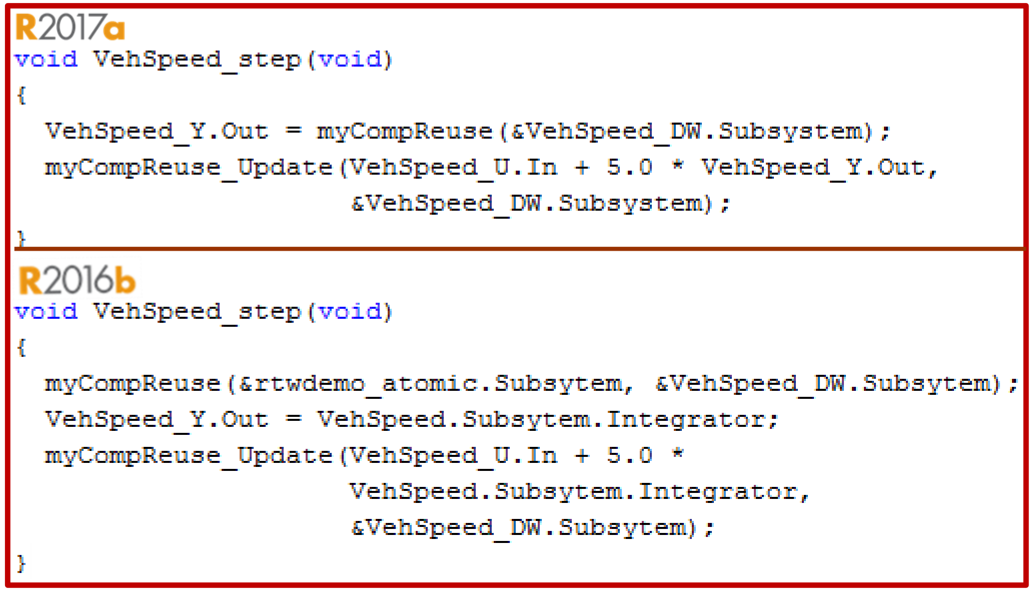
Select the settings for the subsystem bl
code generation, select 'Treat as atomic

Function packaging: Reusable function

Function name options: User specified

Function name:

myCompReuse



R2017a

```
void VehSpeed_step(void)
{
  VehSpeed_Y.Out = myCompReuse(&VehSpeed_DW.Subsystem);
  myCompReuse_Update(VehSpeed_U.In + 5.0 * VehSpeed_Y.Out,
                     &VehSpeed_DW.Subsystem);
}
```

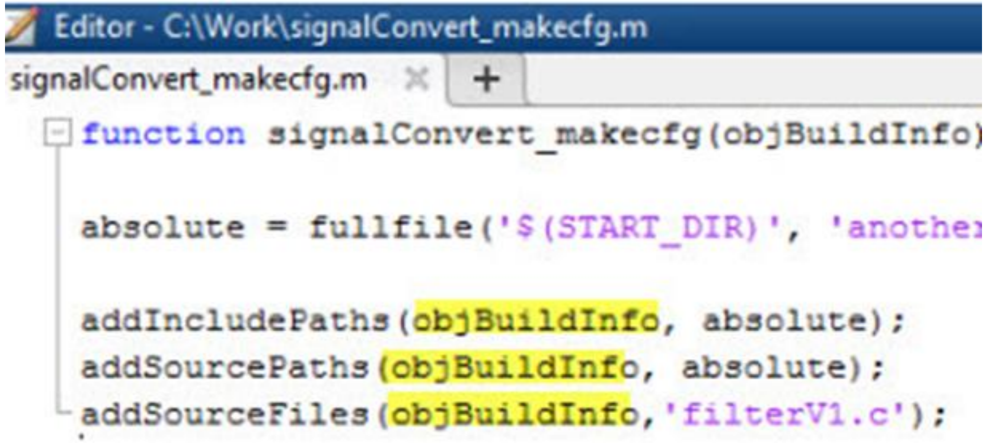
R2016b

```
void VehSpeed_step(void)
{
  myCompReuse(&rtwdemo_atomic.Subsytem, &VehSpeed_DW.Subsytem);
  VehSpeed_Y.Out = VehSpeed.Subsytem.Integrator;
  myCompReuse_Update(VehSpeed_U.In + 5.0 *
                     VehSpeed.Subsytem.Integrator,
                     &VehSpeed_DW.Subsytem);
}
```

Build Process Customization for S-Functions

Customize generated makefiles to integrate with your legacy build processes

- Customize generated makefiles by creating `makecfg.m` and `yourSFunction_makecfg.m` files that use `RTW.BuildInfo` functions to specify:
  - Additional source files and libraries
  - Preprocessor macro definitions
  - Compiler flags

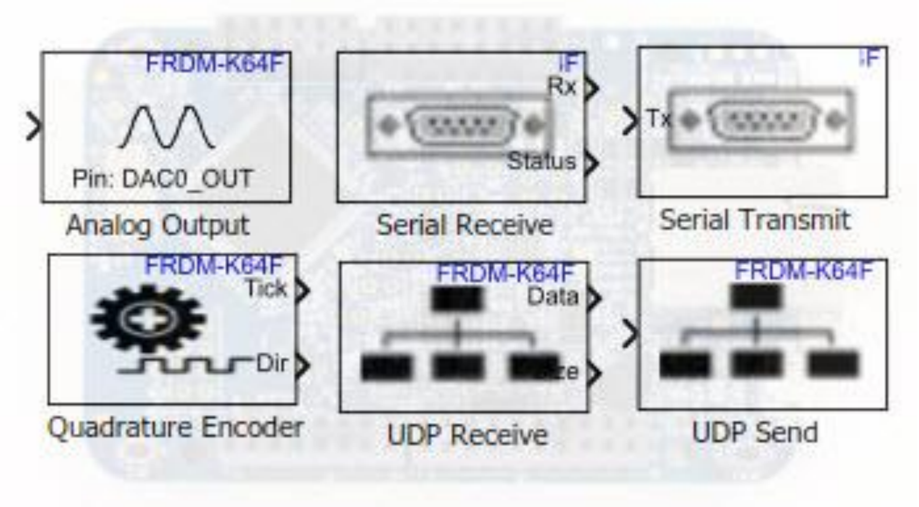


```
function signalConvert_makecfg(objBuildInfo)

absolute = fullfile('$(START_DIR)', 'anothe

addIncludePaths(objBuildInfo, absolute);
addSourcePaths(objBuildInfo, absolute);
addSourceFiles(objBuildInfo,'filterV1.c');
```

NXP FRDM-K64F I/O Blocks

Generate code on NXP FRDM-K64F Board using expanded I/O blocks

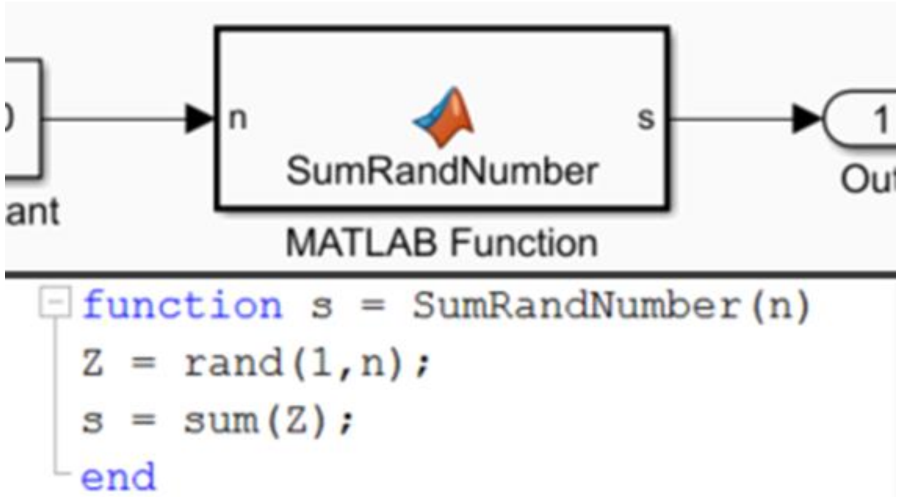
- Easily develop feedback controls applications using
  - Analog Output
  - Quadrature Encoder
  - Serial Receive and Transmit
  - UDP Receive and UDP Send



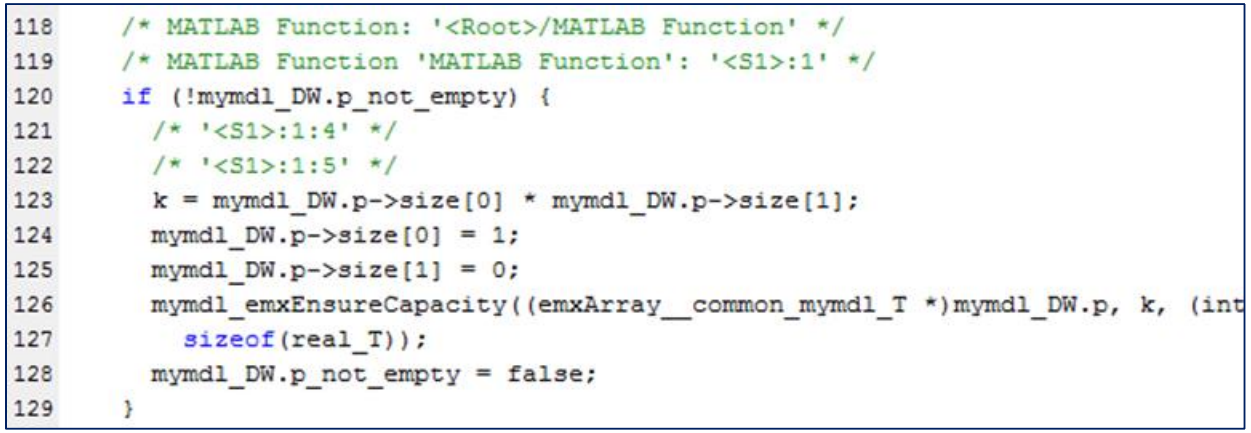
Dynamic Memory Allocation for MATLAB Function Block

Simulate and generate C code for embedded vision, automated systems, and compute intensive software that needs dynamic memory

- Dynamically allocated data generated as structure type, emxArray, and include utility manager functions
- Identifiers for emxArrays and utility functions are customizable (with Embedded Coder)



```
function s = SumRandNumber(n)
Z = rand(1,n);
s = sum(Z);
end
```



```
    /* MATLAB Function: '<Root>/MATLAB Function' */
    /* MATLAB Function 'MATLAB Function': '<S1>:1' */
    if (!mymdl_DW.p_not_empty) {
      /* '<S1>:1:4' */
      /* '<S1>:1:5' */
      k = mymdl_DW.p->size[0] * mymdl_DW.p->size[1];
      mymdl_DW.p->size[0] = 1;
      mymdl_DW.p->size[1] = 0;
      mymdl_emxEnsureCapacity((emxArray__common_mymdl_T *)mymdl_DW.p, k, (int
        sizeof(real_T));
      mymdl_DW.p_not_empty = false;
    }
```