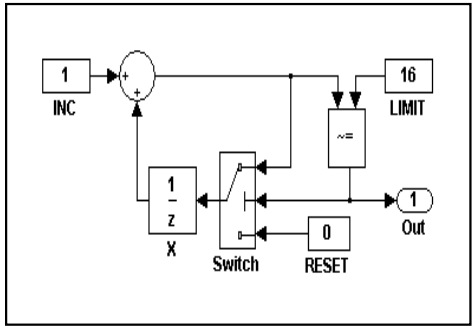
MATLAB EXPO 2019

간편해진 C/C++ 코드 생성방법 소개

유재흥

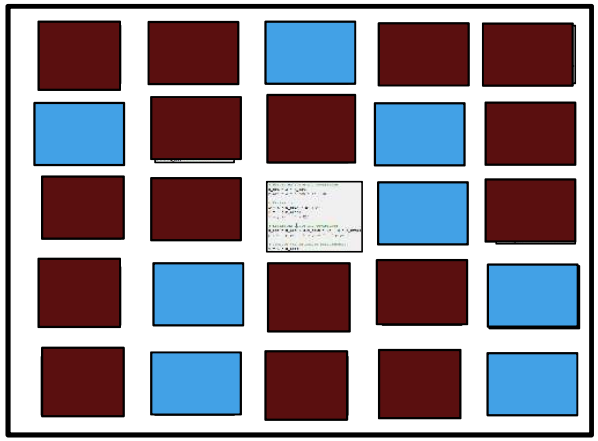
Goal: Generating Perfect Code for Your Environment

Model



Code Generation

Application Framework



- **Function Interfaces**
- **Data Definition**
- **Data Access**

- **How to do these?**
- **Any easy way to do these?**

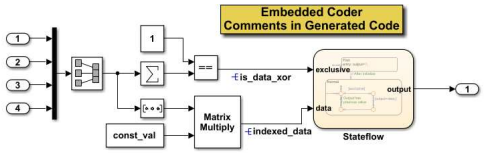
Agenda

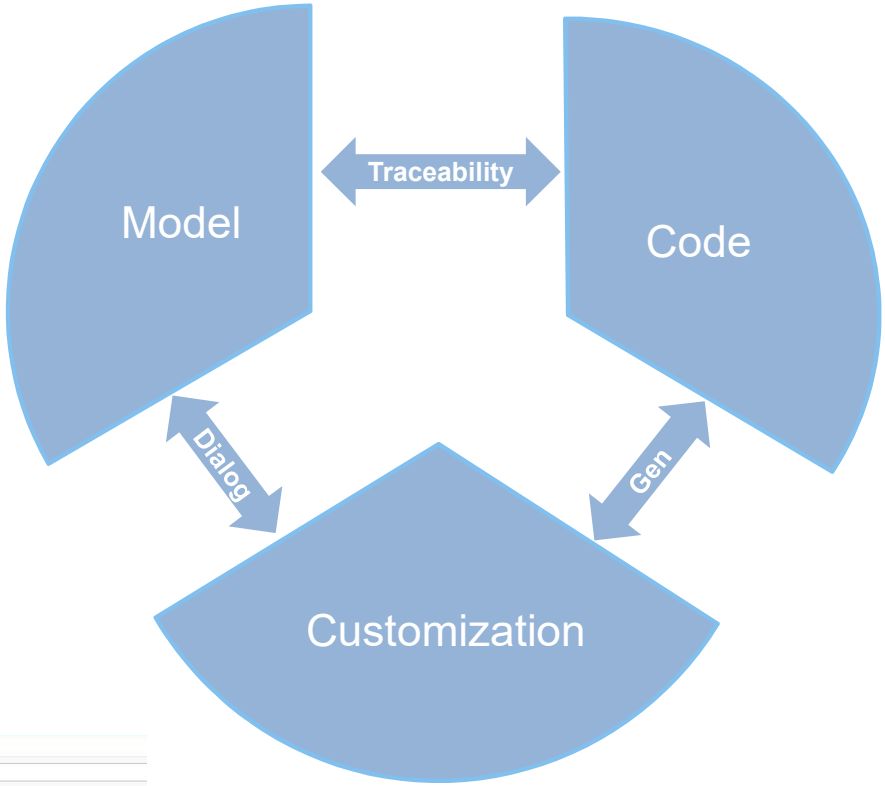
- A Brand New Way to think about Customization
- A Brand New Way to interact with the Code
- Data Access Customization
- Row Major and Multi-Dimensional Indexing

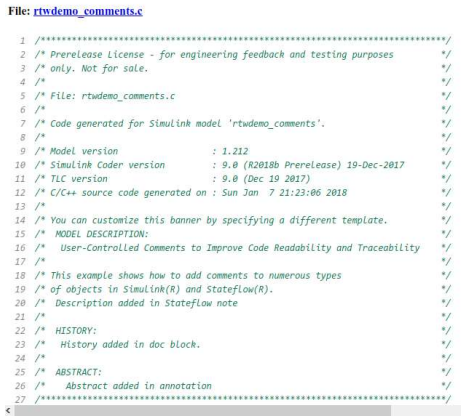
Agenda

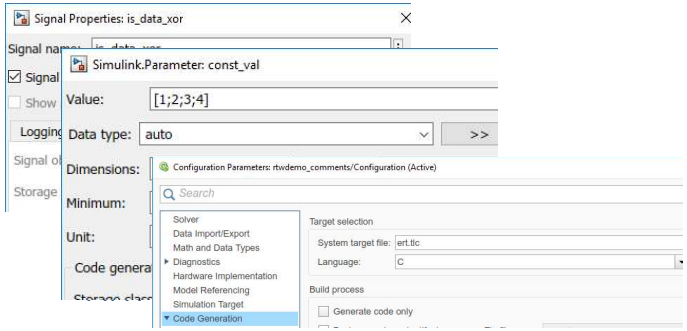
- A Brand New Way to think about Customization
- A Brand New Way to interact with the Code
- Data Access Customization
- Row Major and Multi-Dimensional Indexing

Code Customization Workflow (Before R2018A)

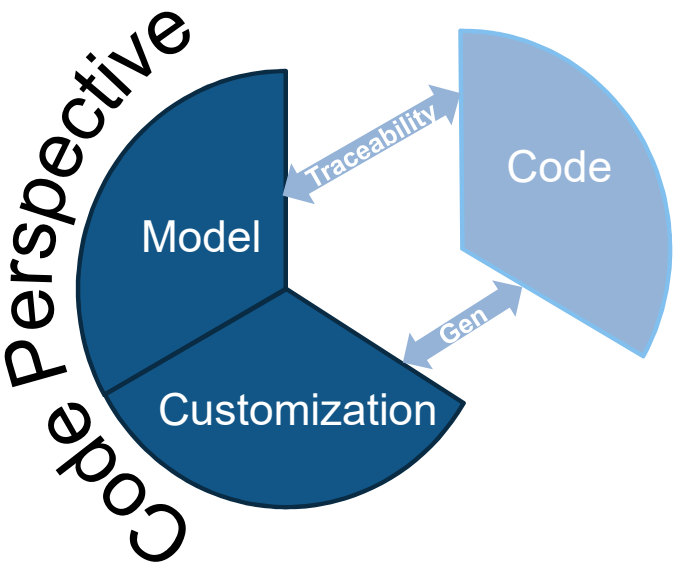


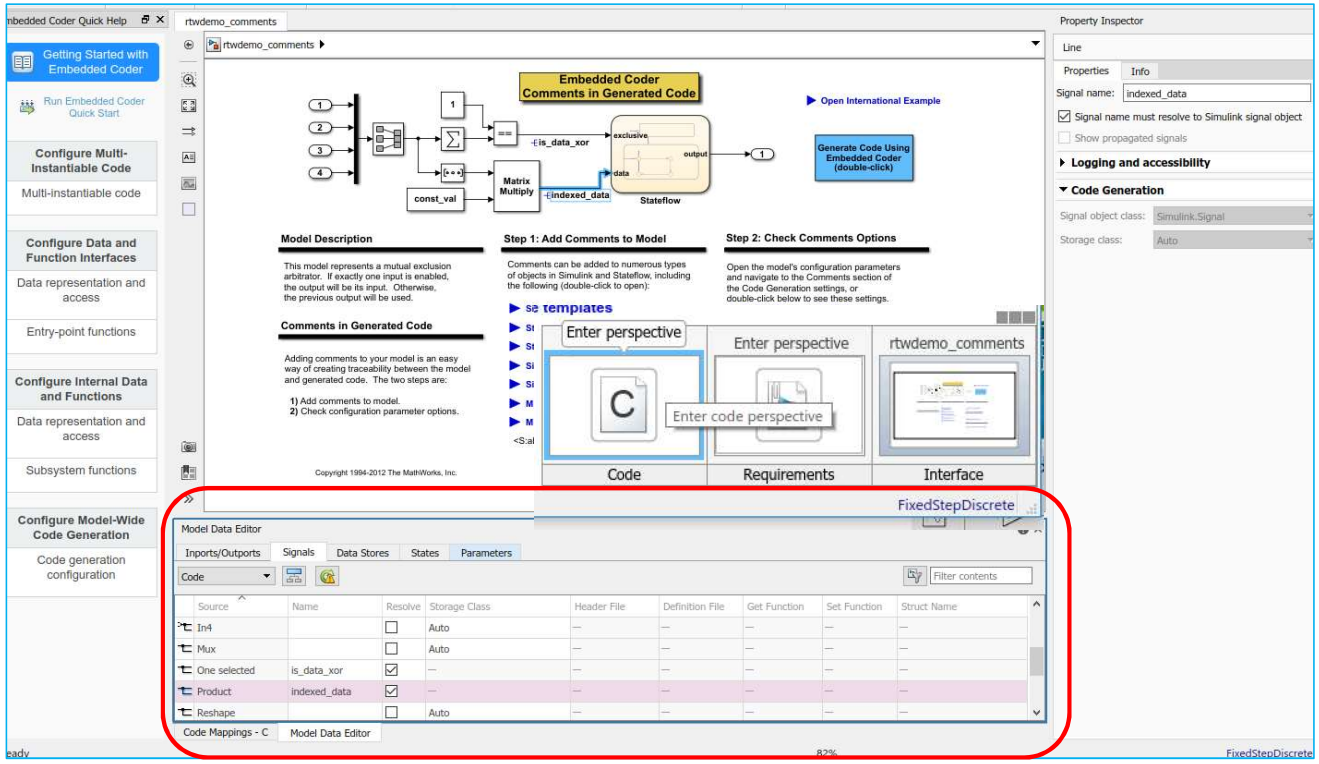




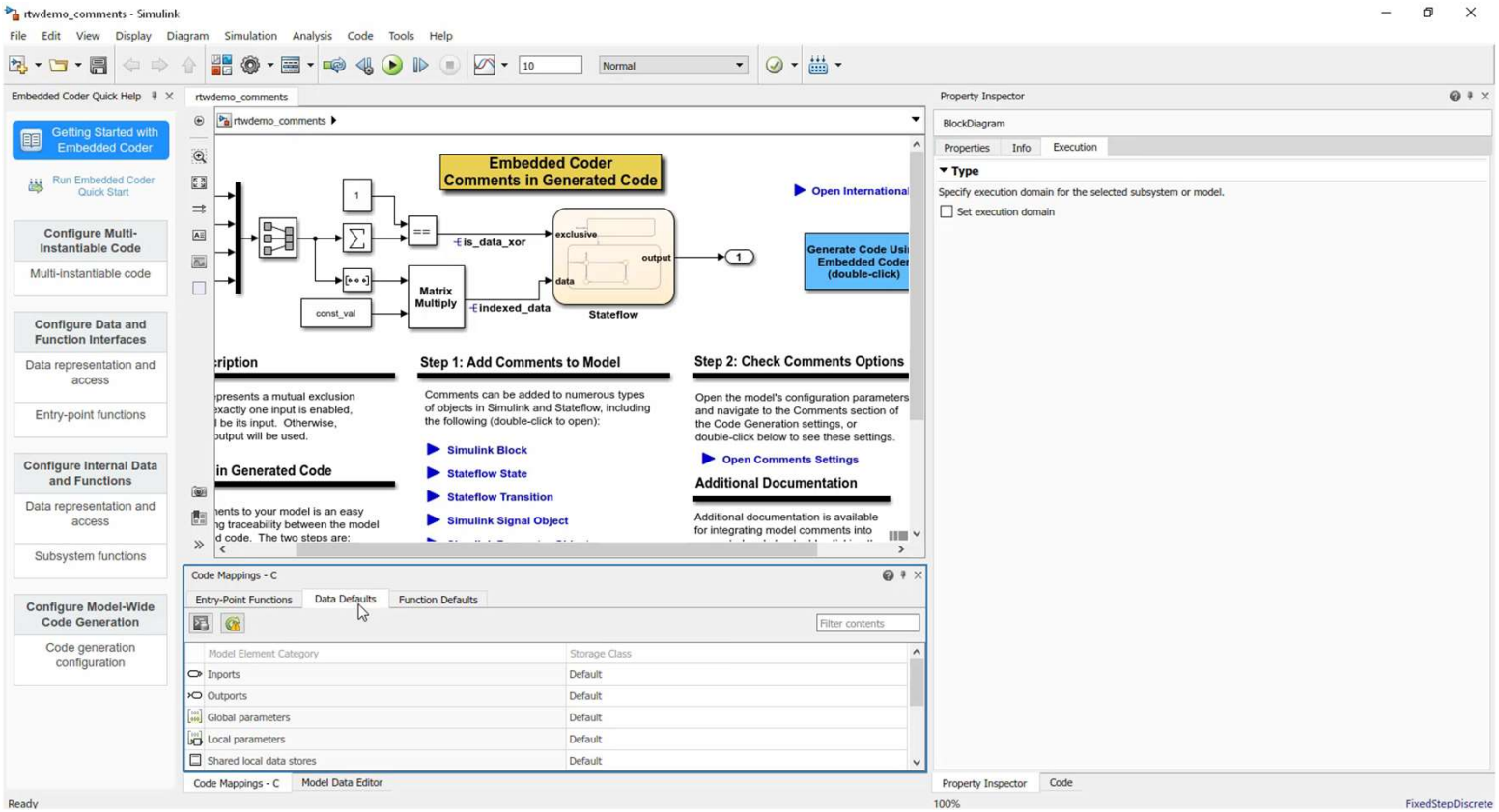


Code Customization Workflow with Code Perspective (R2018a)



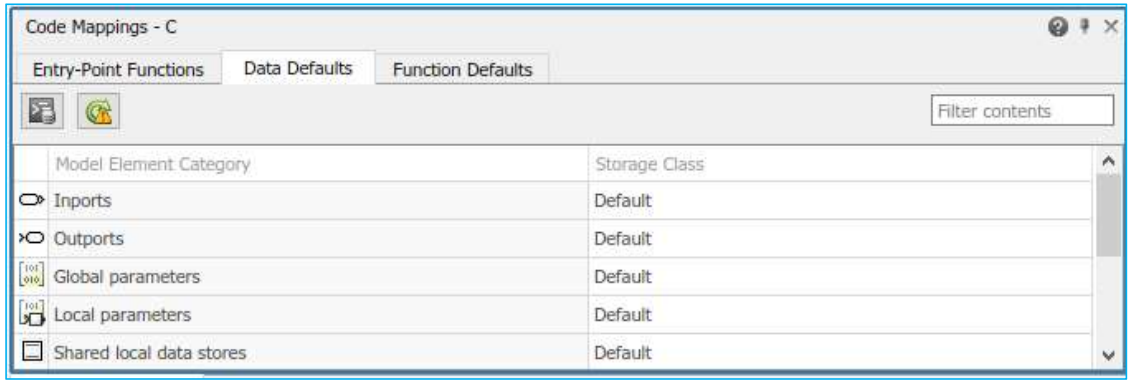


Overview of Code Inspector



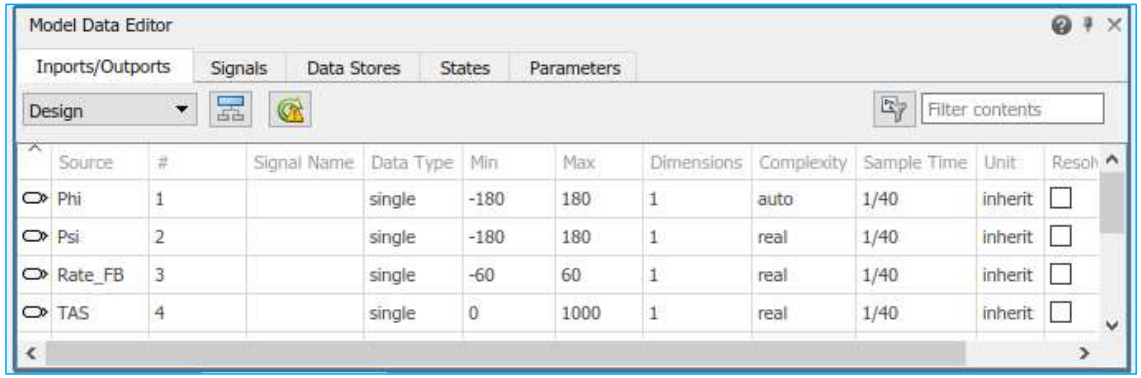
Customization Spreadsheets

- Code Mappings allow for default specification of functions and variables



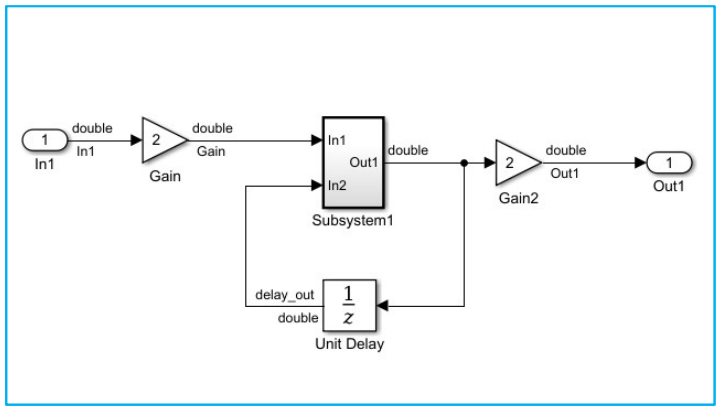
Model Element Category	Storage Class
Inports	Default
Outports	Default
Global parameters	Default
Local parameters	Default
Shared local data stores	Default

- Model Data Editor allows for individual element customization

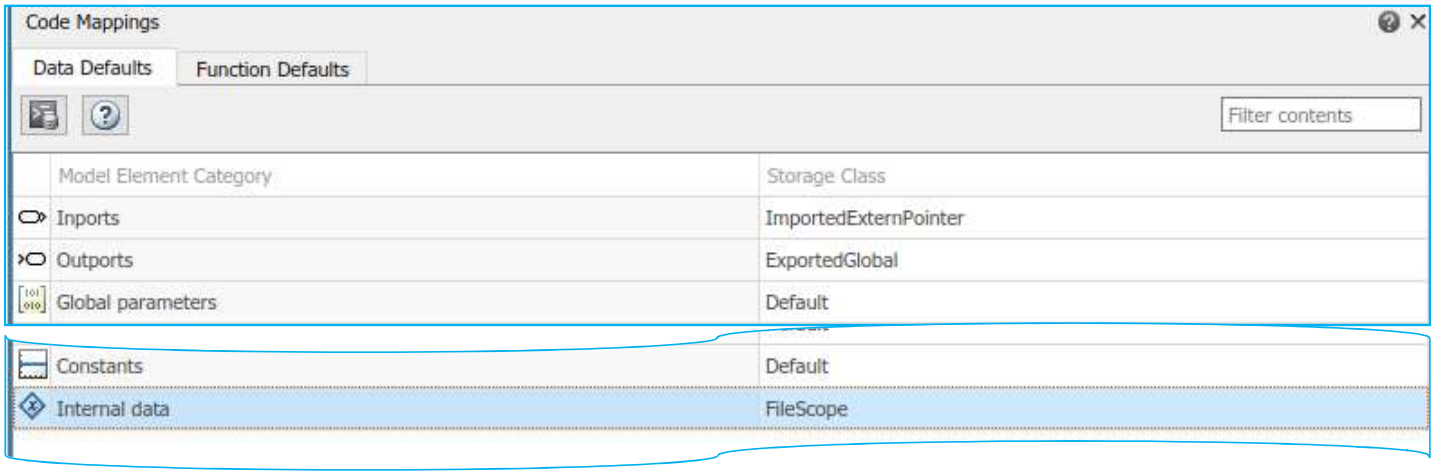


Source	#	Signal Name	Data Type	Min	Max	Dimensions	Complexity	Sample Time	Unit	Resol
Phi	1		single	-180	180	1	auto	1/40	inherit	☐
Psi	2		single	-180	180	1	real	1/40	inherit	☐
Rate_FB	3		single	-60	60	1	real	1/40	inherit	☐
TAS	4		single	0	1000	1	real	1/40	inherit	☐

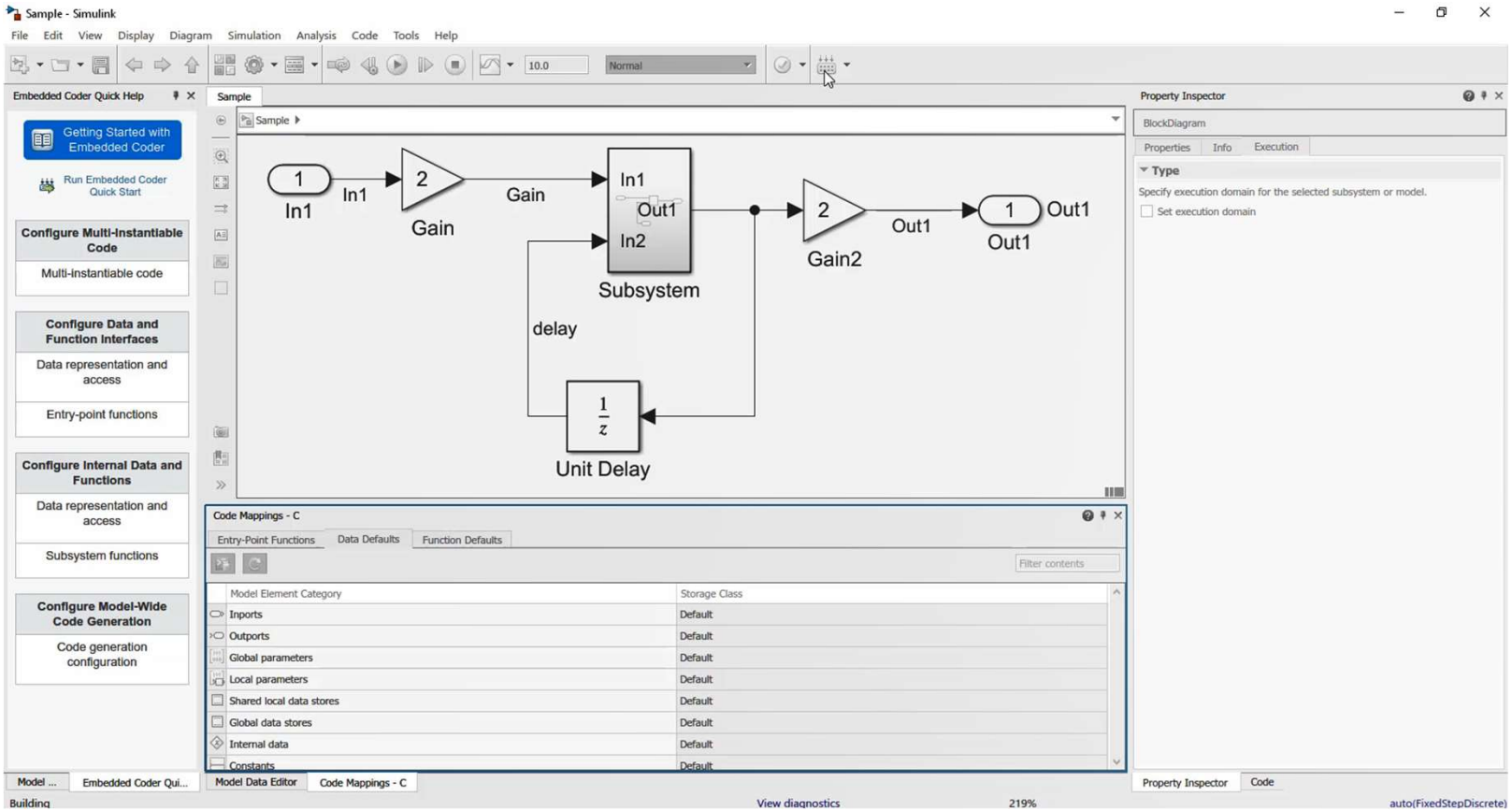
A Case Study




```
extern real_T *In1;
real_T Out1;
static real_T delay;
```

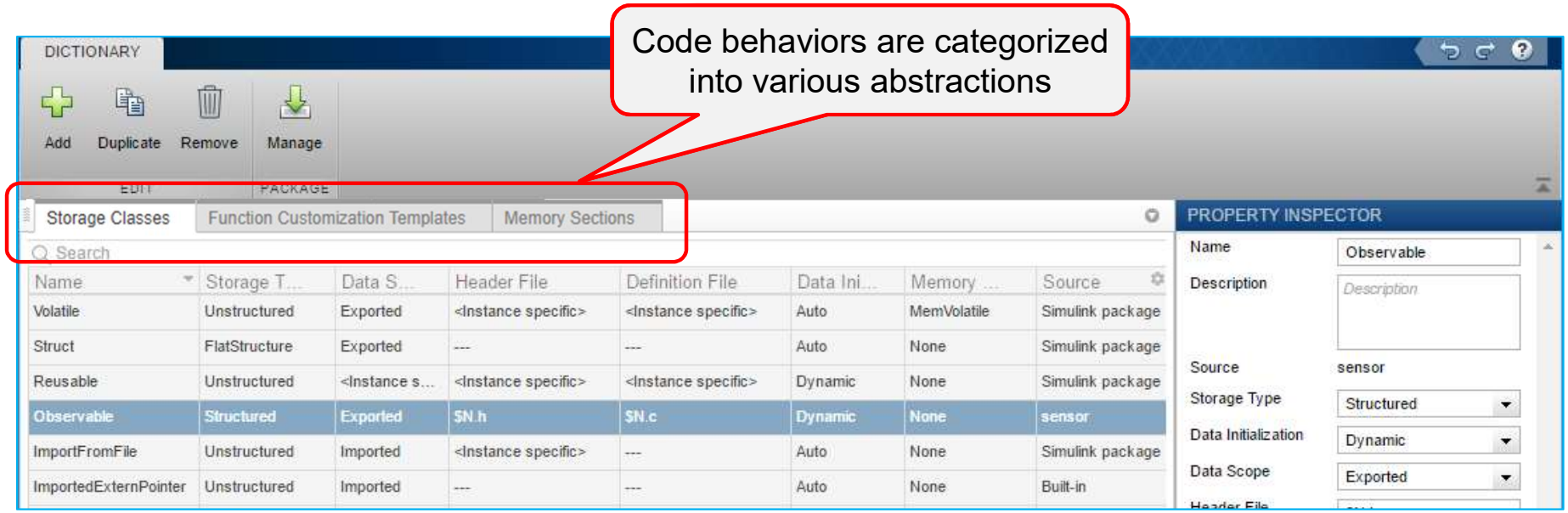


Generating code with C Code mapping



Defining Code Generation Behavior Without Packages and Classes

- Embedded Coder Dictionary



Step to Set Embedded Coder Dictionary

Setting Memory Sections

Storage Classes | Function Customization Templates | Memory Sections

Search

Name	Comment	Pre Statement	Post Statement		
MemConst	/* Const memory section */				Simulink package
MemVolatile	/* Volatile memory section */				Simulink package
MemConstVolatile	/* ConstVolatile memory s...				Simulink package
MyMemorySection		#pragma section near=...	#pragma endsection	Each variable	Sample

PSEUDOCODE PREVIEW

Declaration:

```
#pragma section near=RAM_ASW
extern DATATYPE DATANAME;
#pragma endsection
```

Definition:

```
#pragma section near=RAM_ASW
DATATYPE DATANAME;
#pragma endsection
```

OR

MyMemorySection

Description

Source: Sample

Comment:

Pre Statement: #pragma section near=RAM_ASW

Post Statement: #pragma endsection

Statements Surround: Each variable

Step to Set Embedded Coder Dictionary

Storage Classes | Function Customization Templates | Memory Sections

Search

Name	Comment	Pre Statement	Post Statement		
MemConst	/* Const memory section */				Simulink package
MemVolatile	/* Volatile memory section */				Simulink package
MemConstVolatile	/* ConstVolatile memory s...				Simulink package
MyMemorySection		#pragma section near=...	#pragma endsection	Each variable	Sample

Setting Memory Sections

MyMemorySection

Description

Source: Sample

Comment

Pre Statement: #pragma section near=RAM_ASW

Post Statement: #pragma endsection

Statements Surround: Each variable

PSEUDOCODE PREVIEW

Declaration:

```
#pragma section near=RAM_ASW
extern DATATYPE DATANAME;
#pragma endsection
```

Definition:

```
#pragma section near=RAM_ASW
DATATYPE DATANAME;
#pragma endsection
```

Rich code preview to assist in defining customizations

Step to Set Embedded Coder Dictionary

Storage Classes | Function Customization Templates | Memory Sec

Search

Name	Data Sco...	Header File	Storage			
FileScope	File	---	Unstructur			
Localizable	Auto	---	Unstructured	Auto	None	Simulink package
Struct	Exported	---	FlatStructure	Auto	None	Simulink package
GetSet	Imported	<Instance specific>	AccessFunction	Auto	---	Simulink package
CompilerFlag	Imported	---	Unstructured	Macro	---	Simulink package
Reusable	<Instance spe...	<Instance specific>	Unstructured	Dynamic	None	Simulink package
SignalStruct	Exported	$N.h	Structured	Dynamic	None	Sample
ParamStruct	Exported	$N.h	Structured	Static	None	Sample
MyStorage	Exported	$N.h	Unstructured	Dynamic	MyMemorySection	Sample

Setting for Variables & Functions

PSEUDOCODE PREVIEW

For single-instance data:

Declaration: exported through file: "Sample.h"

```
#pragma section near=RAM_ASW
extern DATATYPE DATANAME;
#pragma endsection
```

Definition: defined in file: "Sample.c"

```
#pragma section near=RAM_ASW
DATATYPE DATANAME;
#pragma endsection
```

For multi-instance data:

PROPERTY INSPECTOR

Name: MyStorage

Description: Description

Source: Sample

File Placement

Data Scope: Exported

Header File: $N.h

Definition File: $N.c

Storage

Storage Type: Unstructured

Data Initialization: Dynamic

Memory Section: MyMemorySection

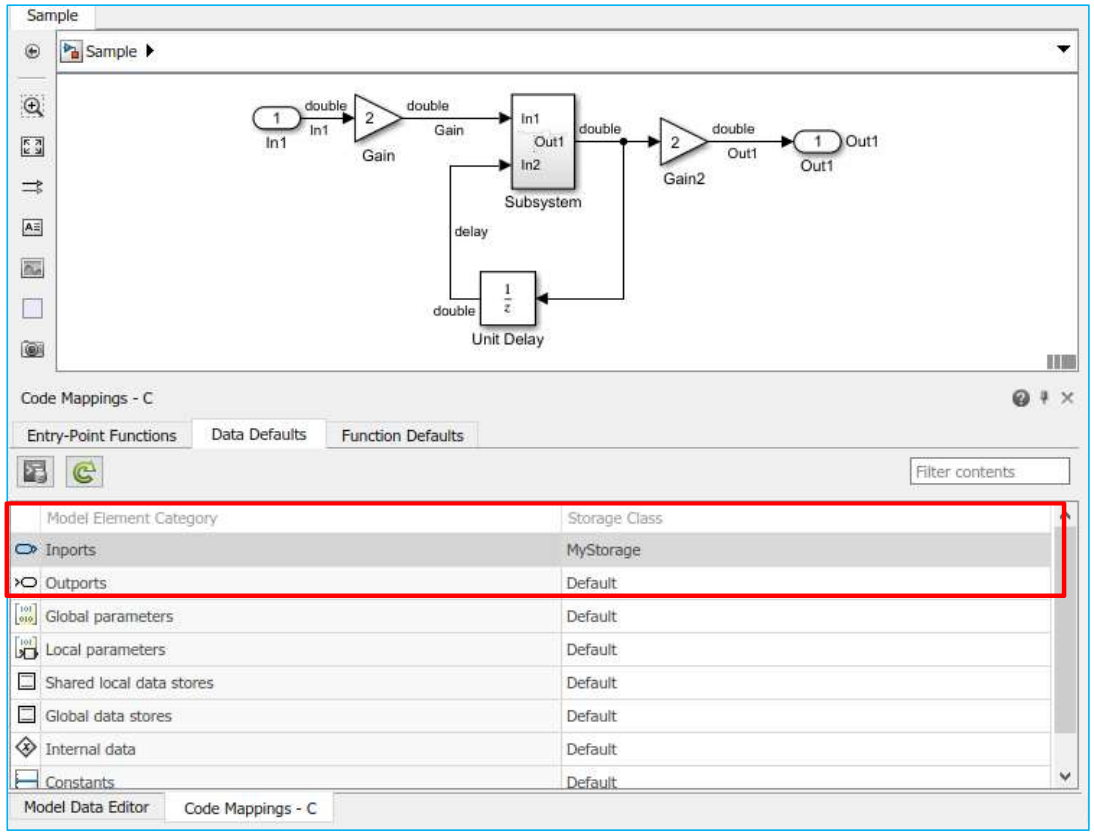
Qualifiers

Allowed Usage

Parameters

Signals

Loading Storage Class from Embedded Coder Dictionary



Configuring C Step function interface

Configure C Step Function Interface: rtwdemo_roll

Configure the generated C Step function interface, including function name and arguments.

C function prototype: arg_Ail_Cmd = rtwdemo_roll_step(* arg_Phi, arg_Psi, arg_Rate_FB, arg_...

C Step Function Name: rtwdemo_roll_step

☑ Configure arguments for Step function prototype

Get default (* invokes update diagram)

C return argument: Ail_Cmd

Port Name	Port Type	C Type Qualifier	C Identifier Name
Phi	Inport	Pointer	arg_Phi
Psi	Inport	Value	arg_Psi
Rate_FB	Inport	Value	arg_Rate_FB
TAS	Inport	Value	arg_TAS
AP_Eng	Inport	Value	arg_AP_Eng
HDG_Mode	Inport	Value	arg_HDG_Mode

OK Cancel Help Apply

Roll Axis Autopilot Model

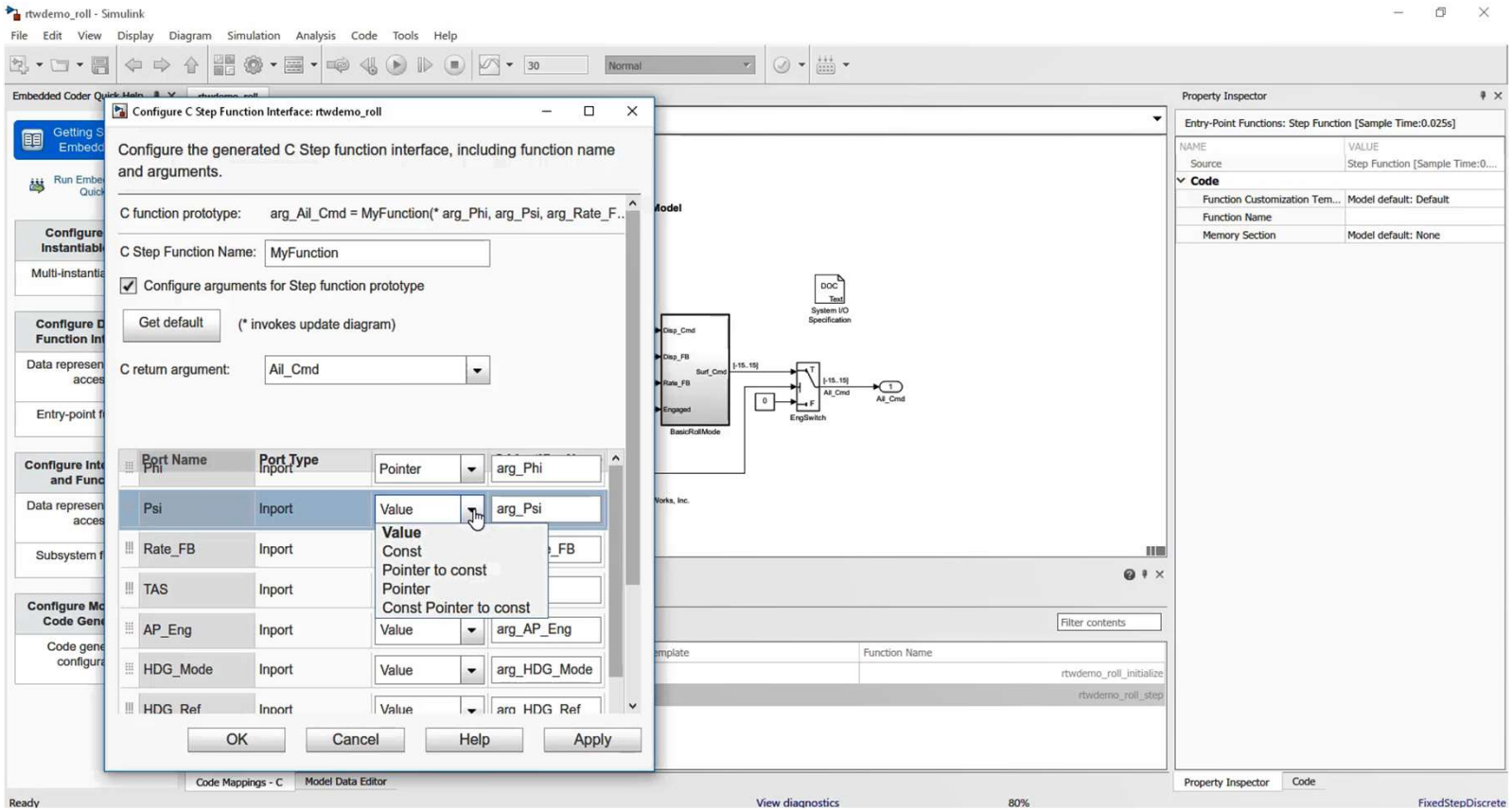
Initialize Function — Model default: Default

Step Function [Sample Time:0.025s] — Model default: Default

Configure Prototype

```
/* Block signals and states (default storage) */
DW rtDW;

/* Model step function */
real32_T rtwdemo_roll_step(real32_T *arg_Phi, real32_T arg_Psi, real32_T
  arg_Rate_FB, real32_T arg_TAS, boolean_T arg_AP_Eng, boolean_T arg_HDG_Mode,
  real32_T arg_HDG_Ref, real32_T arg_Turn_Knob)
{
  boolean_T rtb_NotEngaged_f;
  real32_T rtb_Sum1;

  /* specified return value */
  real32_T arg_Ail_Cmd;

  /* Outputs for Atomic SubSystem: '<Root>/RollAngleReference' */
  /* UnitDelay: '<S7>/FixPt Unit Delay1' */
  rtb_Sum1 = rtDW.FixPtUnitDelay1_DSTATE;

  /* Switch: '<S7>/Enable' incorporates:
   *  Inport: '<Root>/AP_Eng'
   *  Logic: '<S3>/NotEngaged'
   */
  if (arg_AP_Eng) {
```

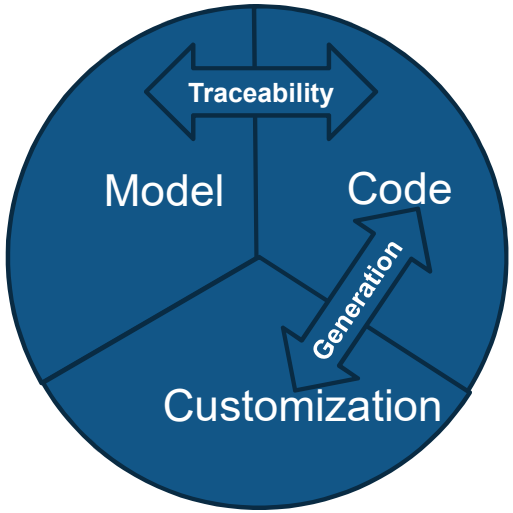
Generating C-Code with C Step function interface

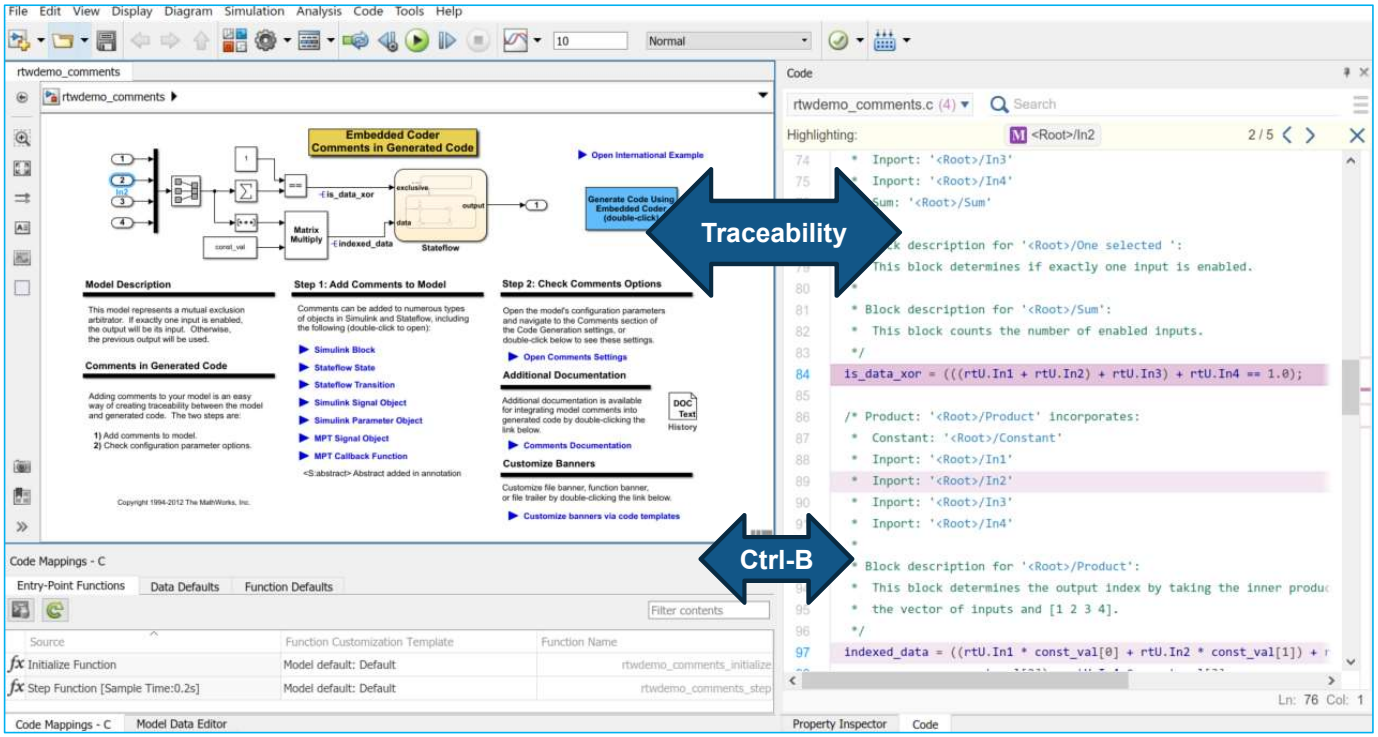


Agenda

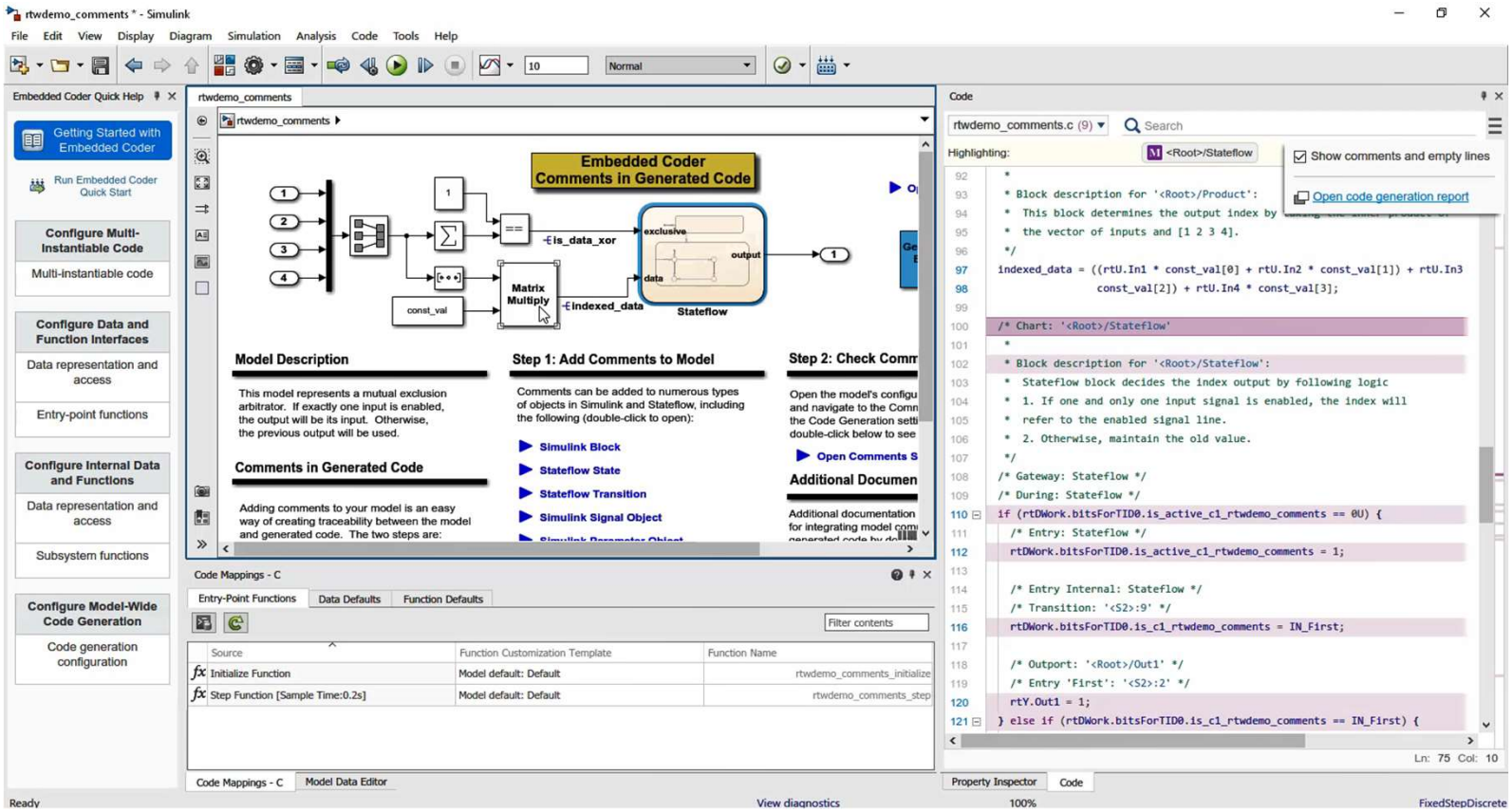
- A Brand New Way to think about Customization
- A Brand New Way to interact with the Code
- Data Access Customization
- Row Major and Multi-Dimensional Indexing

In R2018b we will integrate the last piece, the code





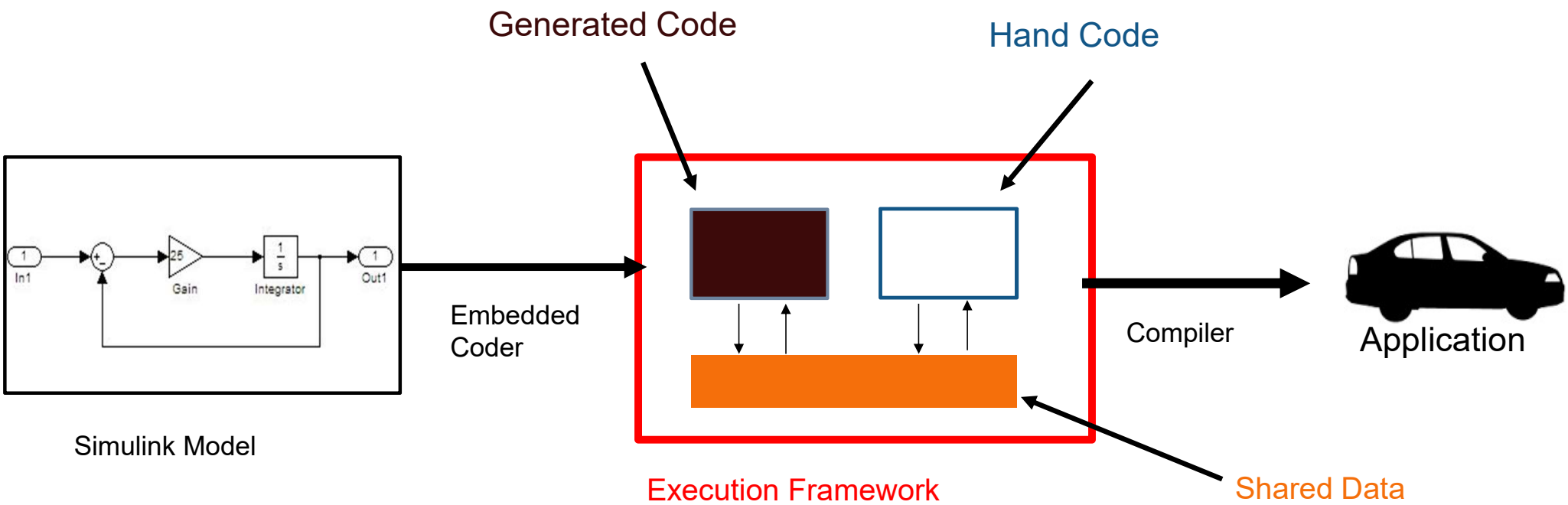
Showing Traceability Model to Code



Agenda

- A Brand New Way to think about Customization
- A Brand New Way to interact with the Code
- Data Access Customization
- Row Major and Multi-Dimensional Indexing

Deployment Scenario: Generated Code as Part of Application



Control *how* data is accessed

- Separate the algorithm code from how data is stored and accessed

```
void model_step()
{
  real_T rtb_Gain;
  rtb_Gain = 2.0 * ExtU.Temperature;
}
```

```
void model_step()
{
  real_T rtb_Gain;
  rtb_Gain = 2.0 * getTemperature();
}
```

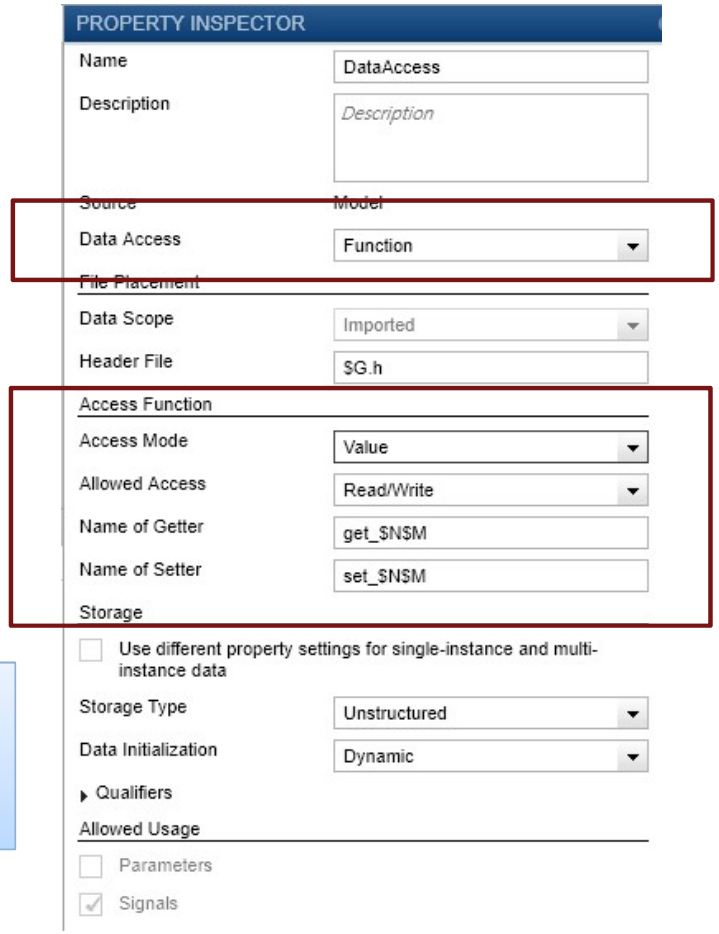
Generated Code

```
real_T getTemperature()
{
  ensureTemperatureDataCorrectness();
  return temperature;
}
```

User Code

Data Access Customization Capabilities in R2019a

- Applicable to Root I/O and Parameters
- Improvements over current Get/Set
  - Export with header file and definition file
  - Richer function name specification
  - Pass by value vs. Pass by pointer




```
real_T getVector(int elem)
{
  return data[elem];
}
```

By value

```
real_T* getVector()
{
  return data;
}
```

By pointer

Agenda

- A Brand New Way to think about Customization
- A Brand New Way to interact with the Code
- Data Access Customization
- Row Major and Multi-Dimensional Indexing

Row Major vs. Column Major

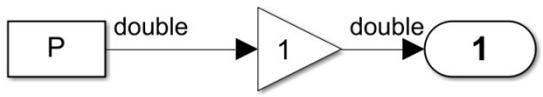
Column major code generation: M[] = {11, 21, 31, 12, 22, 32}; → M(2) = 31 Column-major indexing

MATLAB: $M = \begin{bmatrix} 11 & 12 \\ 21 & 22 \\ 31 & 32 \end{bmatrix}$

Row major code generation: M[] = {11, 12, 21, 22, 31, 32}; → M(2) = **21** Row-major indexing

Row Major Code Generation

$$P = \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}$$



Column major layout

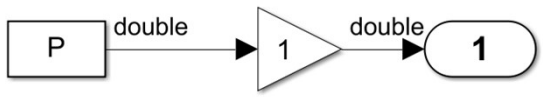
```
P rtP = {
  /* Variable: P
   * Referenced by: '<Root>/Constant'
   */
  { 1.0, 3.0, 5.0, 2.0, 4.0, 6.0 }
};
```

Row major layout

```
P rtP = {
  /* Variable: P
   * Referenced by: '<Root>/Constant'
   */
  { 1.0, 2.0, 3.0, 4.0, 5.0, 6.0 }
};
```

Row Major and Multidimension Indexing

$$P = \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}$$



Row major layout

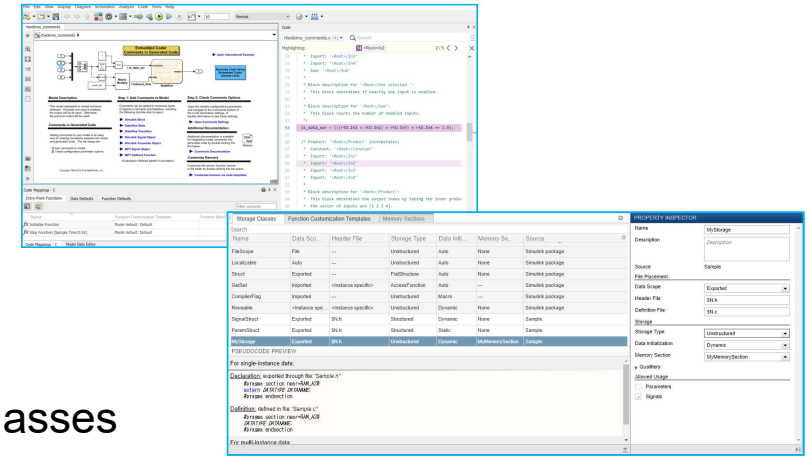
```
P rtP = {
  /* Variable: P
   * Referenced by: '<Root>/Constant'
   */
  { 1.0, 3.0, 5.0, 2.0, 4.0, 6.0 }
};
```

Multi-Dimensional layout

```
P[3][2] = { { 1.0, 2.0 }, { 3.0, 4.0 }, { 5.0, 6.0 } } ;
```

Key Takeaways

- Code Perspective
  - Enable to set Default Code Pattern of Blocks
  - Modifying Block Properties in Spreadsheets
  - Showing Model and Code in same Window
- Embedded Coder Dictionary
  - Defining Code Behavior without Package and Classes
- Data Access Customization
  - Accessing Data with Function
- Row Major Layout and Multi-Dimensional Indexing
  - Supporting Row Major and Multi-Dimensional Indexing



```
void model_step()
{
  real_T rtb_Gain;
  rtb_Gain = 2.0 * ExtU.Temperature;
}
```

```
void model_step()
{
  real_T rtb_Gain;
  rtb_Gain = 2.0 * getTemperature();
}
```

Generated Code

```
real_T getTemperature()
{
  ensureTemperatureDataCorrectness();
  return temperature;
}
```

User Code

Row major layout

```
P rtP = {
  /* Variable: P
   * Referenced by: '<Root>/Constant'
   */
  { 1.0, 3.0, 5.0, 2.0, 4.0, 6.0 }
};
```

Multi-Dimensional layout

```
P[3][2] = { { 1.0, 2.0 }, { 3.0, 4.0 }, { 5.0, 6.0 } } ;
```

MATLAB EXPO 2019

감사합니다