간편해진 C/C++ 코드 생성방법 소개

유재흥

MATLAB EXPO 2019
Goal: Generating Perfect Code for Your Environment

- 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 Data Editor allows for individual element customization
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

Code behaviors are categorized into various abstractions
Step to Set Embedded Coder Dictionary

Setting Memory Sections

PSEUDOCODE PREVIEW

Declaration:

```
#pragma section near=RAM אש
extern DATATYPE DATANE
#pragma endsection
```

Definition:

```
#pragma section near=RAM אש
DATATYPE DATANE
#pragma endsection
```
Step to Set Embedded Coder Dictionary

Setting Memory Sections

Rich code preview to assist in defining customizations
Step to Set Embedded Coder Dictionary

Setting for Variables & Functions
Loading Storage Class from Embedded Coder Dictionary
Configuring C Step function interface
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

Simulink Model

Generated Code

Hand Code

Embedded Coder

Compiler

Execution Framework

Shared Data

Application
Control *how* data is accessed

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

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

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

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

**Generated Code**

**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

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

By value

```c
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:

MATLAB:

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

$M[[]] = \{11, 21, 31, 12, 22, 32\}$; $M(2) = 31$

Column-major indexing

Row major code generation:

MATLAB:

$M[[]] = \{11, 12, 21, 22, 31, 32\}$; $M(2) = 21$

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

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
감사합니다