MATLAB EXPO 2015
KOREA
2015년 5월 21일 목요일
인터컨티넨탈 코엑스, 서울
C/C++ 사용자를 위한 MATLAB 활용: 알고리즘 개발 및 검증

이웅재 부장
Signal Processing Algorithm Design with C/C++
You can **enhance** your **C** and **C++** development process using the data analysis, visualization, algorithm development, code generation, deployment, test and verification capabilities of **MATLAB**
Signal Processing Algorithm Design with C/C++ and MATLAB
In this presentation

- Combining MATLAB and C/C++ for
  - Test and verification
  - Algorithm development, research, design
  - Implementation and deployment

- Examples from
  - Wireless communications
  - Image processing and computer vision
  - Audio and digital signal processing
Section 1: Test and Verification
MATLAB and C/C++

Use Case 1: Call MATLAB from C/C++

- MATLAB Engine Interface
  - Interactive Algorithm Development
  - Debugging of C or MATLAB Code
  - Plotting / Visualization
  - Connect to Data Acquisition, Test and Measurement Systems
Bluetooth Example
Find Engineering Defects in C with MATLAB
Plot and Visualize Data in MATLAB

>> Example
Bluetooth Example
Test and Debug Your C Project and Visualize Data/Results

- **Add** header file and libraries to your project
- **Package** data into MATLAB data structure (mxArray)
- **Use** engine routines to:
  - Send data to MATLAB
  - Perform computations in MATLAB
  - Plot data in MATLAB
- For C++ example, see:
  - [www.codeproject.com/samples/matlabeng.asp](http://www.codeproject.com/samples/matlabeng.asp)
C Engine Routines: Call MATLAB from C

Call MATLAB Engine
Call MATLAB® from C/C++ and Fortran programs

C and Fortran Functions

- engOpen (C and Fortran)
- engOpenSingleUse (C)
- engClose (C and Fortran)
- Engine (C)
- engEvalString (C and Fortran)
- engGetVariable (C and Fortran)
- engPutVariable (C and Fortran)
- engGetVisible (C)
- engSetVisible (C)
- engOutputBuffer (C and Fortran)

engOpen (C and Fortran)
Start MATLAB engine session

C Syntax
#include "engine.h"
Engine *engOpen(const char *startcmd);

Fortran Syntax
mxPointer engOpen(startcmd)
character(*) startcmd

Arguments
startcmd
String to start the MATLAB® process. On Windows® systems, the startcmd string must be NULL

Returns
Pointer to an engine handle, or NULL if the open fails.

Description
This routine allows you to start a MATLAB process for using MATLAB as a computational engine.
Formal C/C++ Code Verification

Polyspace Client (C/C++, Ada)
Polyspace Server (C/C++, Ada)

MATLAB EXPO 2015
Polyspace link on MATLAB Coder

Easily run Polyspace on MATLAB Coder generated code

- Allows to run a polyspace verification on generated C/C++ code generated with MATLAB Coder
- Only needs the code generation folder as entry point to automatically setup and run a Polyspace Code Prover or Bug Finder verification
- View results on generated C/C++ code in Polyspace graphical environment
Do you need to access powerful and trusted ready-to-use signal processing libraries from your C projects?

IIR and FIR filters…

FFT, DFT, IFFT, Power Spectral Density…

Waveform and pulse generation…
Do you need to do advanced signal analysis, measurements and visualization from your C projects?

Settling time, peak to peak, peak to rms, rise time, fall time…

Spectrum analysis, time and frequency domain visualization…
Section 2: Algorithm Development

- Specification
- Algorithm Development
  - C/C++
  - MATLAB
- Specialized Libraries
- Algorithm Development
- Testing & Debugging
- Deployment
Basic Filter Design Example

Filter Design and Analysis App

- Lowpass FIR Filter Design

Impulse response

Pole/Zero plot

>> Example
Signal Analysis, Processing, and Algorithms

- Perform interactive signal processing
  - Make key signal measurements
  - Perform time and frequency analysis
- Design, visualize, and optimize filters
  - Digital IIR and FIR filters
  - Interactive design and analysis tools
- Develop algorithms for signal processing
  - Signal transforms, multi-rate operations
  - Statistical functions, linear prediction
- Utilize MATLAB Apps
  - Signal Analysis
  - Filter Design and Analysis
  - Window Design and Analysis
Real-Time Filter Tuning Example

Develop advanced signal processing algorithms with MATLAB
Utilize specialized ready-to-use libraries

>> Example
Real-Time Filter Tuning Example
Algorithm Development with MATLAB and C/C++

- **Develop** algorithms in an integrated environment
  - MATLAB
  - System objects, scripts, functions
  - Simulation and visualization

- **Perform** iterative testing and debugging
  - Validate that algorithm matches specification
  - Compare different design choices / trade-offs

- **Prepare** your algorithm for implementation or deployment
Streaming and Frame-Based Signal Processing

- **System objects**
  - *MATLAB objects* – encapsulate algorithms and I/O
  - Enable streaming signal processing

- **Advantages**
  - State management, indexing and buffering
  - Computation and memory management
  - Fixed-point arithmetic*, C code generation**

* With Fixed-Point Designer  
** With MATLAB Coder
Signal Processing Applications with MATLAB

- Ready-to-use algorithm libraries
- Enable real-time signal processing applications
  - Stream and frame processing
  - Advanced design options
  - Fixed-point arithmetic

- Digital Signal Processing
- Wireless and Wired Communications
- Computer Vision and Image Processing
- Phased Array and Radar Systems
- RF System Design and Analysis
Do you need equivalent C code for the signal processing algorithms created in MATLAB?

Why?

Integrate with external C projects

Embedded applications

Standalone simulations
Section 3: Implementation and Deployment

- Specification
- Algorithm Development (C/C++ MATLAB)
- Specialized Libraries
- Testing & Debugging
- Deployment

Implementation & Deployment
MATLAB and C/C++

Use Case 2: Generate C code from MATLAB

- MATLAB Coder
  - Generate C from MATLAB
  - Integrate with your C/C++ source
  - Build and deploy complete applications

- No royalty or deployment restrictions on code generated with MATLAB Coder
Lane Markings Example
Generate C from MATLAB and integrate with C/C++
Lane Markings Example
Generate C from MATLAB

- **Prepare** your MATLAB algorithm
  - Make implementation choices
  - Use supported language features

- **Test** if your MATLAB code is compliant
  - Validate that MATLAB program generates code
  - Accelerate execution of user-written algorithm

- **Generate** source code or MEX
  - Iterate your MATLAB code to optimize
  - Implement as source, executable or library
Implementation Considerations

- **MATLAB**
  - Polymorphic data types
  - Automatic memory management
  - Built-in matrix & array support

- **C**
  - Explicit data type declaration & handling
  - Define and manage: Data size, memory, matrices, arrays,

Kalman Filter Example
MATLAB Code: 7 lines
C Code: 107 lines
Supported MATLAB Language Features and Functions

Broad set of language features and functions/system objects supported for code generation

<table>
<thead>
<tr>
<th>Matrices and Arrays</th>
<th>Data Types</th>
<th>Programming Constructs</th>
<th>Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Matrix operations</td>
<td>• Complex numbers</td>
<td>• Arithmetic, relational, and logical operators</td>
<td>• MATLAB functions and subfunctions</td>
</tr>
<tr>
<td>• N-dimensional arrays</td>
<td>• Integer math</td>
<td>• Program control</td>
<td>• Variable-length argument lists</td>
</tr>
<tr>
<td>• Subscripting</td>
<td>• Double/single-precision</td>
<td>(if, for, while, switch)</td>
<td>• Function handles</td>
</tr>
<tr>
<td>• Frames</td>
<td>• Fixed-point arithmetic</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Persistent variables</td>
<td>• Characters</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Global variables</td>
<td>• Structures</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Numeric class</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Variable-sized data</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• MATLAB Class</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• System objects</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Supported algorithms
• More than 1100 MATLAB operators, functions, and System objects for:
  • Communications
  • Computer vision
  • Image processing
  • Phased Array signal processing
  • Signal processing
  • Statistics
Supported Functions

- Aerospace Toolbox
- Communications System Toolbox
- Computer Vision System Toolbox
- DSP System Toolbox
- Image Processing Toolbox
- Phased Array System Toolbox
- Signal Processing Toolbox
- Statistics Toolbox

1100+ Supported Functions
MATLAB and C/C++

Use Case 3: Reuse your C/C++ libraries in MATLAB

- **loadlibrary**
  - Load your library functions
  - Access your IP as custom libraries in MATLAB
  - Combine with standard MATLAB functions, scripts, System objects
MATLAB and C/C++
Use Case 4: Reuse your C/C++ code in MATLAB

- `mex`
  - Compile and reuse your C/C++ code in MATLAB
  - Combine with standard MATLAB functions, scripts, System objects

Examples (OpenCV)
MATLAB and C/C++
Use Case 5: Reuse MATLAB IP in your C/C++ code

- Create specialized algorithm libraries in MATLAB
  - Generate C with MATLAB Coder
  - Or, automatically create shared libraries with MATLAB Compiler or MATLAB Coder
    - DLL, LIB
  - Access MATLAB algorithms from C/C++ with DLLs or LIBs

No royalty or deployment restrictions

>> Example (lane)
Deploying Applications with MATLAB

1. MATLAB Application
2. MATLAB Compiler
3. .dll

MATLAB Desktop

End-User Machine

MATLAB Compiler Runtime (MCR)
MATLAB Compiler/SDK and MATLAB Coder

MATLAB Coder

MATLAB Compiler

MATLAB EXPO 2015
## MATLAB Compiler and MATLAB Coder

<table>
<thead>
<tr>
<th>Feature</th>
<th>MATLAB Compiler</th>
<th>MATLAB Coder</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Output</strong></td>
<td>Executable or software component/library</td>
<td>Portable and readable C source code</td>
</tr>
<tr>
<td><strong>MATLAB language support</strong></td>
<td>Full</td>
<td>Subset</td>
</tr>
<tr>
<td><strong>Additional libraries</strong></td>
<td>MATLAB Compiler Runtime (MCR)</td>
<td>None</td>
</tr>
<tr>
<td><strong>Supported toolboxes</strong></td>
<td>Most toolboxes</td>
<td>Some toolboxes</td>
</tr>
<tr>
<td><strong>License model</strong></td>
<td>Royalty-free</td>
<td>Royalty-free</td>
</tr>
<tr>
<td><strong>Extensions</strong></td>
<td>MATLAB Compiler SDK</td>
<td>Embedded Coder</td>
</tr>
</tbody>
</table>
MATLAB and C/C++: Summary of Use Cases

- Call MATLAB from C
- Generate C from MATLAB
- Use C Libraries in MATLAB
- Use C Code in MATLAB
- Generate IP libraries or deploy applications from MATLAB
You can enhance your C and C++ development process using the data analysis, visualization, algorithm development, code generation, deployment, test and verification capabilities of MATLAB

MATLAB and C/C++: The Perfect Combination for Signal Processing
Summary

- Develop algorithms in C/C++ and MATLAB
  - MATLAB enhances your C/C++ environment
  - Wide variety of functions and tools available within MATLAB product family

- Move quickly to implementation
  - MATLAB Coder: ANSI C Code
  - MATLAB Compiler and Compiler SDK: Deployable Applications
  - Fixed-Point support

- Integrate with hardware and your C/C++. Distribute freely.
  - No need for a MATLAB license
  - No royalties for generated code
Any Questions ?