MATLAB to C Made Easy
From Research to Implementation

Jean-Philippe Villaréal
Application Engineer
Applications Engineering Group
MathWorks Benelux
June 11, 2015
The Challenge

“Developing advanced, smart products requires a very high level of cooperation between research engineers, and software engineers who have different requirements for their working environment”
Traditional Workflow for Software Development

**REQUIREMENTS**

**RESEARCH & DEVELOPMENT**
- Explore and discover, **graphically display data**
- Gain insight into the problem, **develop algorithms**
- Evaluate options and trade-offs, **quickly iterate**

**IMPLEMENTATION**
- **Embedded C, C++**
- **VHDL, Verilog**
- **.dll, .lib, .exe**
- **MCU**, **DSP**
- **FPGA**, **ASIC**
- **Desktop Applications**

**INTEGRATION**

**TEST & VERIFICATION**
Why Do Engineers Translate MATLAB to C?

- **Implement** C code on processors, or hand off C code to a Software Engineering group
- **Integrate** MATLAB algorithm in an existing C-based environment
- **Prototype** your algorithm on desktop
- **Accelerate** your MATLAB algorithm using MEX
Challenges with Manual Translation from MATLAB to C

- Need to maintain 2 separate implementations
- Coding errors are introduced
- Time consuming and expensive
- Hard to change requirements during the implementation phase
From Research to Implementation

- **MATLAB** is a high-level language for design exploration and to develop applications.

Isn’t the conversion unambiguous?

- **C Code** is well suited for optimizing performance, memory, and processing power.
Points for Debate

- Polymorphism
- Memory allocation
- Processing arrays and matrices
- Graphics and users interfaces

6 Lines of MATLAB
107 Lines of C
A Modern Approach Based On Automation

MATLAB Coder

Fixed-Point Conversion  Disabled

The MATLAB Coder workflow generates standalone C and C++ code from MATLAB code. To begin, select your entry-point function(s).

Generate code for function: Enter a function name
What is the MATLAB Coder?

- C code generation tool
- Based on a *subset of the MATLAB language*
- Enables you to optimize the generated C code to reduce the *footprint* or improve the *performance*
- Suitable for *desktop* and *embedded* applications
Benefits of Automatic C Code Generation

- Maintain only 1 design
- Design faster and get to C quickly
- Less time developing low-level C code
- Focus on improving MATLAB algorithm
- Test more frequently and systematically
Using MATLAB Coder: 3-Step Workflow

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

- **Test** if your MATLAB code is compliant
  - Can my MATLAB program generate code?
  - Test your functionality with MEX
  - Reuse your testbench

- **Generate** source code or keep MEX
  - Iterate your MATLAB code to optimize
  - Implement as source, executable or library
A Versatile C Code Generation Tool
Fixed-Point Conversion

- Integrated workflow to convert to fixed-point at build time
- Full control of fixed-point properties
- Logging of true dynamic range
- Advise on word length
- Results comparison
- Full support for coder
Success Stories
Challenge
Develop and implement an acoustic respiratory monitoring system for wheeze detection and asthma management

Solution
Develop algorithms for detecting wheeze and ambient noise in MATLAB, and use MATLAB Coder to generate code from the algorithms for mobile devices and a web server

Results
- Manual coding effort reduced
- Algorithm development iterations accelerated
- Code maintenance overhead reduced

"MATLAB enables us to rapidly develop, debug, and test sound-processing algorithms, and MATLAB Coder simplifies the process of implementing those algorithms in C. There's no other environment or programming language that we could use to produce similar results in the same amount of time."

Mark Mulvey
iSonea

The AirSonea device, which connects to an asthma patient's smartphone and communicates with wheeze analysis algorithms on iSonea's server.
VivaQuant Accelerates Development and Validation of Embedded Device for Ambulatory ECG Sensing

Challenge
Design and implement an embedded system for extracting accurate information from noisy electrocardiogram signals

Solution
Use MATLAB to develop an algorithm for removing in-band noise, and use Fixed-Point Designer and MATLAB Coder to implement it on an ARM Cortex-M series processor

Results
- Development accelerated by 300%
- Power and memory consumption minimized
- Rigorous testing enabled

“MATLAB, MATLAB Coder, and Fixed-Point Designer enabled our small team to develop a complex real-time signal processing algorithm, optimize it to reduce power and memory requirements, accelerate embedded code implementation, and perform the rigorous testing required for medical device validation.”

Marina Brockway
VivaQuant

ECG snippet before and after processing with VivaQuant’s embedded in-band noise removal algorithm.
Take-aways

- MATLAB Coder makes engineers more productive
  - One design to maintain
  - Reuse of testbench
  - Faster and less error-prone (e.g. 300% productivity gain at VivaQuant)

- 3-step workflow: Prepare – Test – Generate

- Control the look and feel of the C code
  - Cosmetic preferences
  - Processor architecture
  - Optimize for speed or memory
Thank You for Attending!